University of Windsor
Scholarship at UWindsor

Electronic Theses and Dissertations | Theses, Dissertations, and Major Papers

1-1-1987

The application of multi-valued logic to the implementation of Residue Number System Hardware.

Peter D. Bird
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Bird, Peter D., "The application of multi-valued logic to the implementation of Residue Number System Hardware." (1987). *Electronic Theses and Dissertations*. 6805.
https://scholar.uwindsor.ca/etd/6805

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

NOTE TO USERS

This reproduction is the best copy available.

UMI®

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The Application of Multi-Valued Logic to the Implementation of Residue Number System Hardware

by
Peter D. Bird

A Thesis
Submitted to the Faculty of Graduate Studies through the Department of Electrical Engineering in Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science at the
University of Windsor

Windsor, Ontario
April, 1987.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: EC54794

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.
In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform EC54794
Copyright 2010 by ProQuest LLC
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

© Peter Bird 1987
All Rights Reserved.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Abstract

A thorough survey of recent multi-valued circuit techniques was carried out. Based on the findings from this survey, and on the restrictions imposed by the IC fabrication process that was available, a ternary voltage mode logic system was chosen for a more detailed analysis.

To augment the data obtained from the literature survey, three RNS adders were designed, two using standard binary logic and one using the ternary system mentioned above. These designs followed the same basic structure, each comprised a ROM, decoders and latches. The better of the two binary designs was able to produce the sum of two five bit numbers, using modular addition, in 24ns. The ternary design can sum two three trit numbers in 52ns. The impressive speed performance of these circuits is due to the pipelined structure of the designs.

From these design efforts, and those of the authors referred to in the literature survey, it was possible to compare the area, speed and power consumption of the multi-valued and binary approaches to circuit design. In general, it can be stated that multi-valued designs are slower and consume more power than the corresponding binary designs. For many designs, however, especially those based chiefly on ROMs, the multi-valued approach can produce significant area savings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Acknowledgments

The author would like to acknowledge the guidance and support provided by Dr.G.A.Jullien and Dr.W.C.Miller. The ideas and suggestions by other members of the VLSI Design Group were also greatly appreciated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table of Contents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Tables

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

I. INTRODUCTION

In the never ending search for continued improvement in the performance of integrated circuits, many researchers have considered the possibility of using, as an alternative to binary logic, a logic system based on more than two levels. In fact, since 1971 an international symposium on multi-valued logic has been held creating a forum for the expression of ideas and insights into this somewhat obscure field.

The lure of reducing the signal interconnect between components of systems and of increasing the circuit density has spurred many to invest a great deal of time and effort investigating various methods of putting the theoretical ideas into practice.

The basis for the optimism about the potential benefits of multi-valued logic is rather simple. For any given signal in the binary system, the corresponding signal in a multi-valued system would be able to represent three, four or even more logic levels. Consequently, to represent the same information one would need fewer signal lines. Furthermore, there would be fewer instances of the logic needed to process the signals due to the reduced number of signal lines.

The main objective of this thesis is to examine multi-valued logic and evaluate its usefulness when applied to RNS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

hardware structures. The approach that has been taken is illustrated in Fig. 1. A survey of literature on the subject of multi-valued logic will provide valuable information about multi-valued designs and the various techniques that have been explored. In order to verify the results from the literature, a comparison between a multi-valued design and a binary design will also be conducted.

A. Survey of Multi-Valued Logic Techniques

As with standard binary logic, multi-valued logic can be broken down into two main classifications: voltage mode logic and current mode logic. What follows is a discussion of some of the circuits that have been presented in recent literature. Although perhaps somewhat tedious at times, it is felt that this section is necessary to bring the reader up to date on what research is being conducted at other institutions.

1. Voltage Mode Logic

This logic system is the one with which people are most familiar. It functions by representing each logic level with a different voltage. In the binary case, for example, logic 0 might be represented by zero volts and logic 1, by five volts. MOS based circuitry, for the most part, falls into this category due to the voltage controlled nature of the devices themselves. A review of the MOS devices and the basic equations governing their operation is provided in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

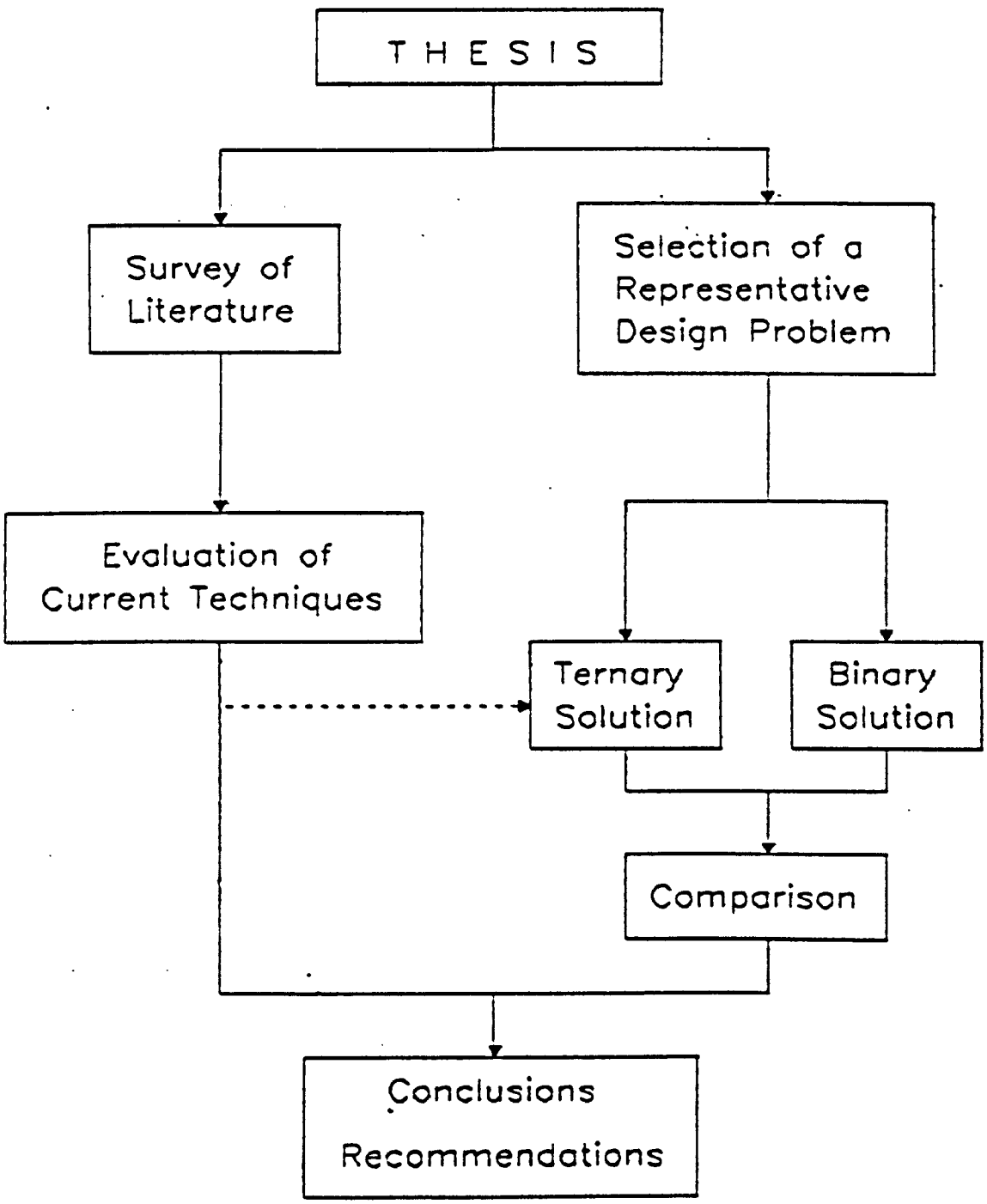


FIG. 1 Approach of Thesis

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix I.

The research in the area of multi-valued voltage mode logic has, in general, been restricted to systems of three and four levels. Beyond that, the problems brought on by the reduced noise margins of the circuits become quite severe rendering them all but useless.

In order to describe the operation of the voltage mode systems, the notation outlined in [HuSa83] has been adopted. The set of available logic levels is defined as

$$S=\{0,1,\ldots,p-1\}$$

where p=number of logic levels

One defines two additional sets, R for the set of permissible output voltages, and T for the set of threshold voltages.

$$R=\{r_0,r_1,\ldots,r_k\}$$
$$T=\{t_1,t_2,\ldots,t_k\}$$

where $t_j \leq t_{j+1}$ and $k,j < p$

Finally, the function

$$U_T^R(e) = \begin{cases} r_0 & \text{if } e < t_1 \\ r_j & \text{if } t_j \leq e < t_{j+1} \\ r_k & \text{if } t_k \leq e \end{cases} \quad (1)$$

is defined where e is a real value representing the input voltage. $U_T^R(e)$ describes the ideal transfer function of a multi-valued circuit operating on a single input signal, e. As an example, if we use this notation to describe the operation of a standard binary CMOS inverter we have the following:

Vdd = 5 volts, Vss = 0 volts
p = 2, S={0,1}, R={5,0}, T={2.5}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Now

$$U_T^R(e) = \begin{cases} 5 & \text{if } e < 2.5 \\ 0 & \text{if } 2.5 \leq e \end{cases}$$

and $Vss \leq e \leq Vdd$

This terminology provides a well structured description of the threshold dependent circuits, however, for general use it can be somewhat unwieldy. To make things clearer, a short hand notation is also introduced. With this notation we simply indicate the set of output logic values which corresponds to a given set of input logic values. Once again taking the binary inverter as an example, we would say:

$$U'(0,1) = (1,0)$$

In the course of discussing the circuits that have been put forth in recent literature, a number of logic functions will be used to illustrate the circuits' operation. These circuits fall into one of two categories; literals and multiplexers or t-gates. Of all the possible sets of literals [McC180], only two are of interest to us. These are:

$$X^i \geq 1 \quad \text{if } X=i$$
$$= 0 \quad \text{if } X \neq i$$

and

$$X^{i,j} \geq 1 \text{ if } X=i \text{ or } X=j$$
$$= 0 \text{ otherwise}$$

where the values of X are elements of S.

Multiplexers allow one of several inputs to pass to the output depending on the value of a control variable. For example, in the binary case:

$$T(X_0, X_1, S) = \begin{cases} X_0 & \text{if } S=0 \\ X_1 & \text{if } S=1 \end{cases}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Most of the work relating to the development of circuits that implement the above logic functions has been concentrated in the direction of ternary systems. Quaternary systems have tended to deal with ROM based designs that avoid the need for actual combinatorial type logic.

a. Ternary Systems

The basic element of the ternary system is the trit. The word comes from ternary digit just as bit comes from binary digit. In the ternary system, there are a number of standard literals which are used to demonstrate whether or not a given circuit technique can be made operational. They are:

Vin	X^0	X^1	X^2	$X^{0,1}$
0	+	0	0	+
1	0	+	0	+
2	0	0	+	0

where + means non zero, ie. 1 or 2.

Specific names have been given to $X^{0,1}$ depending upon what non zero values it takes:

PTI, positive ternary inverter U'(0,1,2) = (2,2,0)
STI, standard ternary inverter U'(0,1,2) = (2,1,0)
NTI, negative ternary inverter U'(0,1,2) = (2,0,0)

Apart from the STI, the remaining literals can be made to convert a ternary input signal into a binary signal, S'=(0,2). Thus after processing the incoming ternary signal one can use normal binary circuits to operate on the output of the literals. This technique has been described in [McC182] as the BITLON (Binary Implemented Ternary Logic

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Network) approach to circuit design. As it turns out, practically all of the ternary designs presented to date, have followed this structure. For the purposes of describing the various techniques only a subset of the above literals will be shown.

In [HuSm83] and [MoHe84], a ternary system was proposed based on a CMOS structure with certain restrictions on the threshold and supply voltages. In Fig. 2, the circuit diagram is presented. If the input signal is equal to Vdd(Vss) then only the nMOS(pMOS) device will be on and the output will be pulled to Vss(Vdd). If, however, the input is equal to 0 volts, then neither of the transistors will be on and the output will be pulled, in the case of the STI, to ground, by the resistor.

The use of the resistor, of course, leads to static power dissipation in two of the states whereas in standard CMOS, this is avoided. In addition, when either of the transistors is conducting, the gate to source voltage is on the order of 0.5 volts. This severely limits the amount of current that the transistor can conduct. As a consequence, the circuit will suffer from longer delay times, compared to standard CMOS, when trying to charge or discharge nodal capacitances. Despite these drawbacks, the authors in [MoHe84] give a brief description of a functional ternary computer based on this ternary system.

NMOS has been investigated for its possible use in multi-valued applications as shown in [McCl80]. The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Vdd = 1 volt

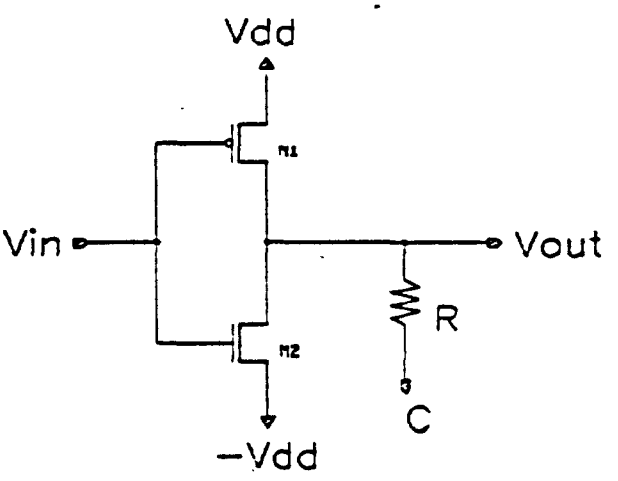
Vdd < |Vt| < 2Vdd

P = 3

S = {0,1,2}

R = {+1.0,0,−1.0}

T = {−0.5,+0.5}



Vout / Vin	NTI C = −Vdd	STI C = 0 volts	PTI C = Vdd
0	2	2	2
1	1	0	2
2	0	0	0

Figure 2. Ternary System Using Resistors

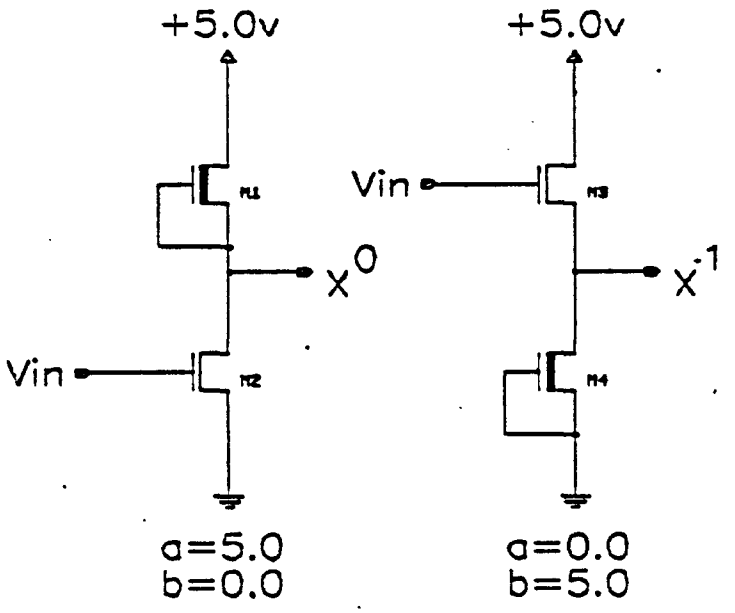
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

circuits presented in this paper, see Fig. 3, require a fabrication process capable of producing both enhancement and depletion mode transistors as well as transistors with different threshold voltages. The operation of these two circuits is fairly straight forward. The depletion mode transistors, m1 and m4, are connected as resistors and consequently pull the outputs to the appropriate supply as long as the input voltages are less than the threshold voltages of the corresponding enhancement mode transistors, m2 and m3. Once the threshold voltages are exceeded, then the outputs, X^0 and X^1, are determined by the ratio of the resistances of m1 to m2 and m3 to m4 respectively.

Another technique which also requires depletion mode devices as well as different threshold voltages was put forward in both [BaAn84] and [HeMo84]. A static ternary inverter designed according to this technique is shown in Fig. 4a. In the former article, the pti and nti were designed without the need for depletion mode devices. These circuits are shown in Fig. 4b. If one examines the restrictions on the threshold voltages one sees once again, that the gate to source voltages will be quite small.

The final voltage mode circuit technique to be mentioned was presented in [MoGa84]. The basis for these designs, see Fig. 5, is the standard CMOS inverter. Only enhancement mode devices are required and a single threshold suffices for the p and n MOS devices. By modify the length and width of the standard inverter different functions can be implemented. These circuits do not suffer from the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



P = 2

S = {0,1,2}

R = {a,0.0,b}

T = {0.7,3.0}

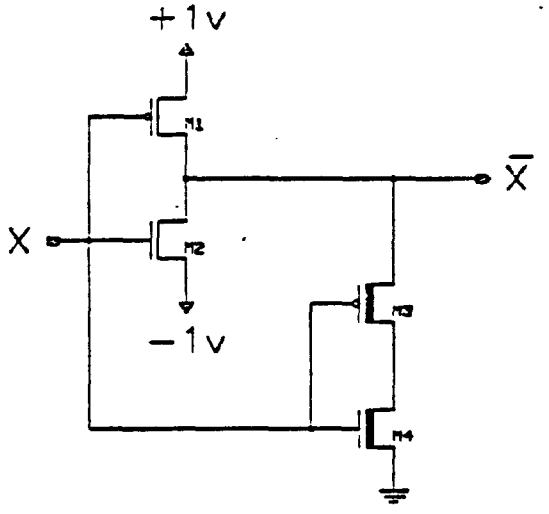
For M3: Vt=3.0v

M2: Vt=0.7v

Vin \ Vout		X^0	X^1
0	0.0v	2	0
1	2.5v	0	0
2	5.0v	0	2

Figure 3. NMOS Ternary System

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



a) STI

P = 2

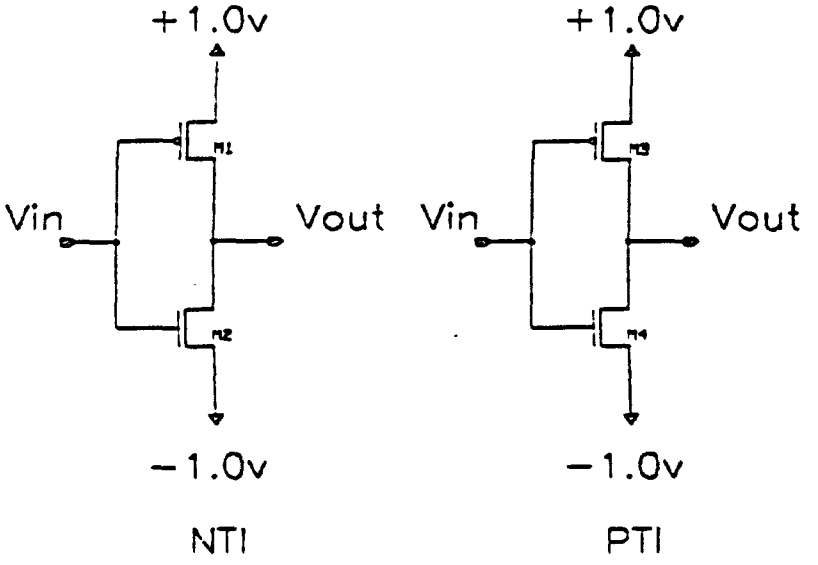
S = {0,1,2}

R = {+1.0,0.0,−1.0}

T = {−0.5,0.5}

Threshold Voltages	
M1	−1.5v
M2	1.5v
M3	0.5v
M4	−0.5v

X	M1	M2	M3	M4	X̄
−1.0	on	off	on	off	1.0
0.0	off	off	on	on	0.0
1.0	off	on	off	on	−1.0

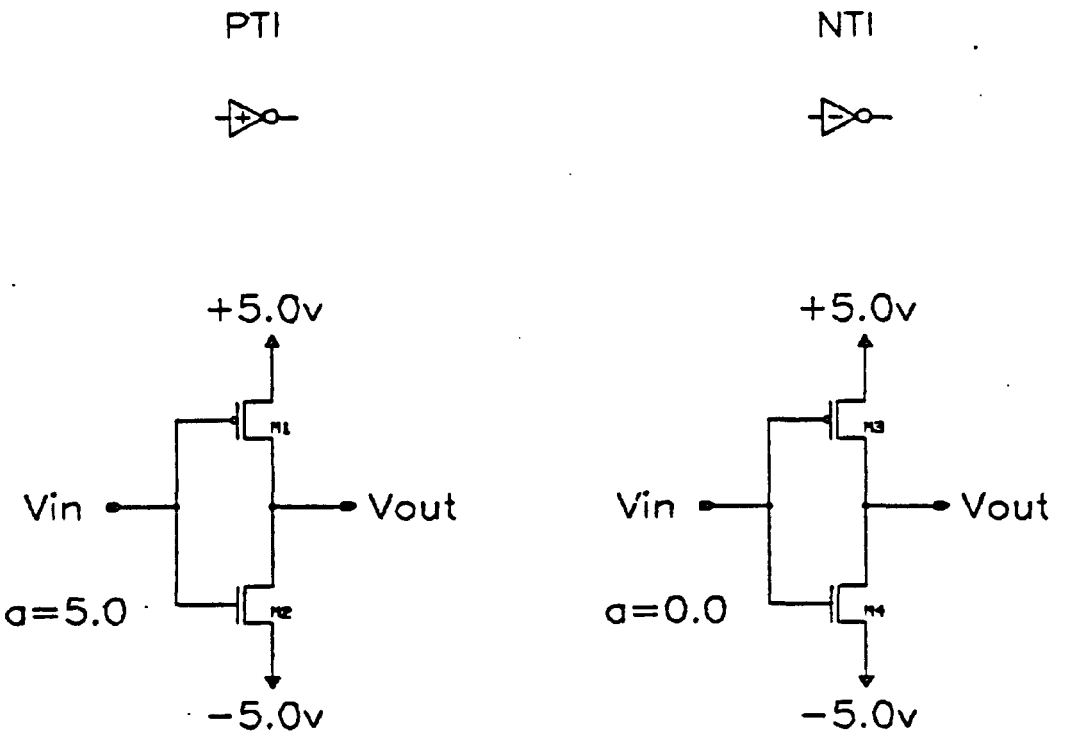


b)

Threshold Voltages	
M1	−0.5v
M2	1.5v
M3	−1.5v
M4	0.5v

Figure 4. CMOS Ternary System

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



P = 2

S = {0,1,2}

R = {5.0,a,-5.0}

T = {-2.5,1.0}

Transistor Sizing (W/L)	
M1	9/5
M2	5/25
M3	9/25
M4	5/5

Threshold Voltages	
M1 and M3	-0.8v
M2 and M4	0.7v

Figure 5. Modifed CMOS Ternary System

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

reduced gate to source voltages that some of the previous designs have suffered from, however, when the input is at 0 volts, there is static power dissipation. A more detailed analysis of these circuits will be done later on since this technique is the one that was used in the design of the ternary adder.

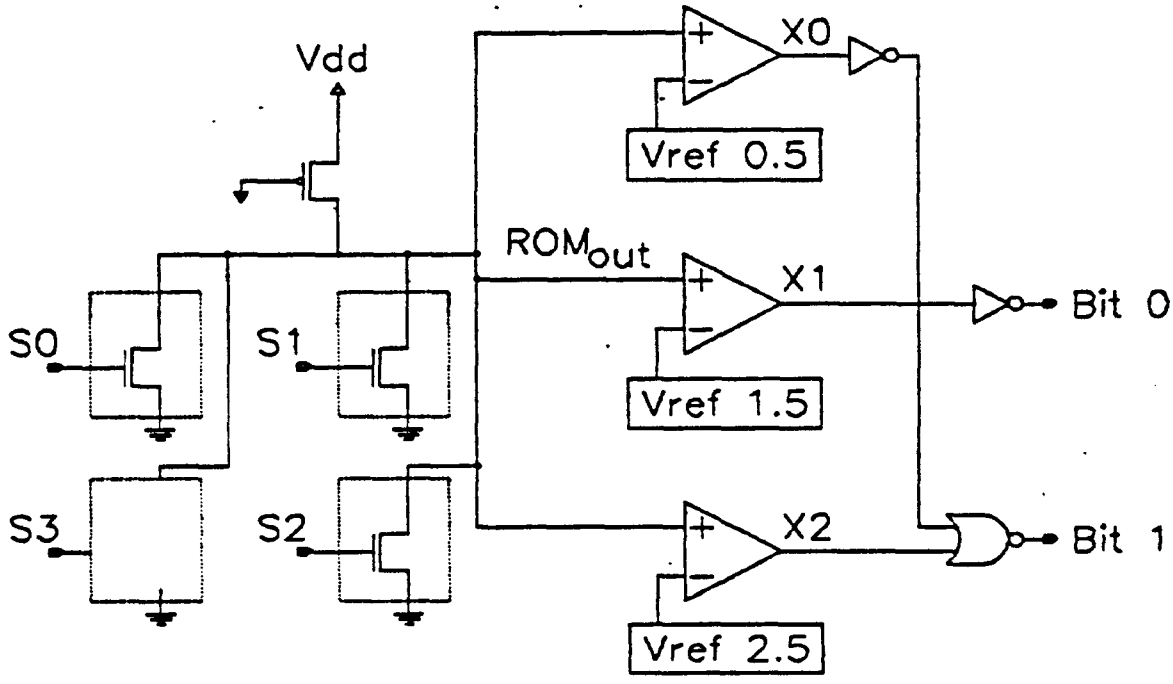
b. Quaternary Systems

In [Star81] the author provides the reader with an insight into the design of a quaternary ROM. In this system, shown in Fig. 6, a given ROM cell location is used to store two bits of information as opposed to just one bit. This is done by varying the impedance of the cell transistor, ie. by changing the width of the gate. Each of the four possible states is represented by a different impedance. The advantage of this approach is that the extra bit of information can be stored in the ROM cell with little or no increase in the area occupied by the cell.

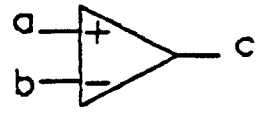
On the output of the ROM, instead of a standard sense amplifier, a set of three comparators is used. The outputs from these comparators are then fed through a simple logic network resulting in two bits of output. The comparators check the ROM cell voltage against a reference voltage with the output being high if the cell voltage is greater than the reference voltage.

If the 4 logic levels are designated as 0,1,2 and 3, then the reference voltages correspond to logic levels

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Transistor Sizing (W/L)	
S0	9/5
S1	7/5
S2	5/5
S3	no tran



c=5v if a>b
c=0v if a<b

S0	S1	S2	S3	ROM_{out}	X0	X1	X2	Bit 0	Bit 1
5	0	0	0	0	0	0	0	5	0
0	5	0	0	1	5	0	0	5	5
0	0	5	0	2	5	5	0	0	5
0	0	0	5	5	5	5	5	0	0

Figure 6. Quaternary Rom Structure

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

0.5, 1.5 and 2.5. Each site transistor, controlled by the selection signals S0 through S1 represents a different logic level.

Although this circuit is indeed multi-valued there is a fundamental difference between it and the circuits that have been discussed thus far. This circuit, which has found applications in real world products, namely Intel's 8087 and 43203 chips, was not intended to operate in a multi-valued environment. The idea was not to reduce the interconnect between sections of the system but rather to take advantage of the greater storage density of a quaternary ROM. The reason this point needs to be emphasized is that this approach means the incoming address lines for the ROM are decoded from binary to the appropriate row and column select signals. In a design that must receive multi-valued inputs, the decoding must convert the multi-valued signals to the binary row and column select signals, which is considerably more involved.

A number of authors have expanded on the work described by M.Stark. For the most part, the techniques remain the same, however some of the details of the transistor sizing and the comparator circuitry have been modified. The reader is referred to [LaEt84], [AgPu84] and [EtNa85] for more information.

An alternate approach to the one just mentioned was presented in [RiNa84]. In this article, the authors describe a ROM in which the four states are represented by

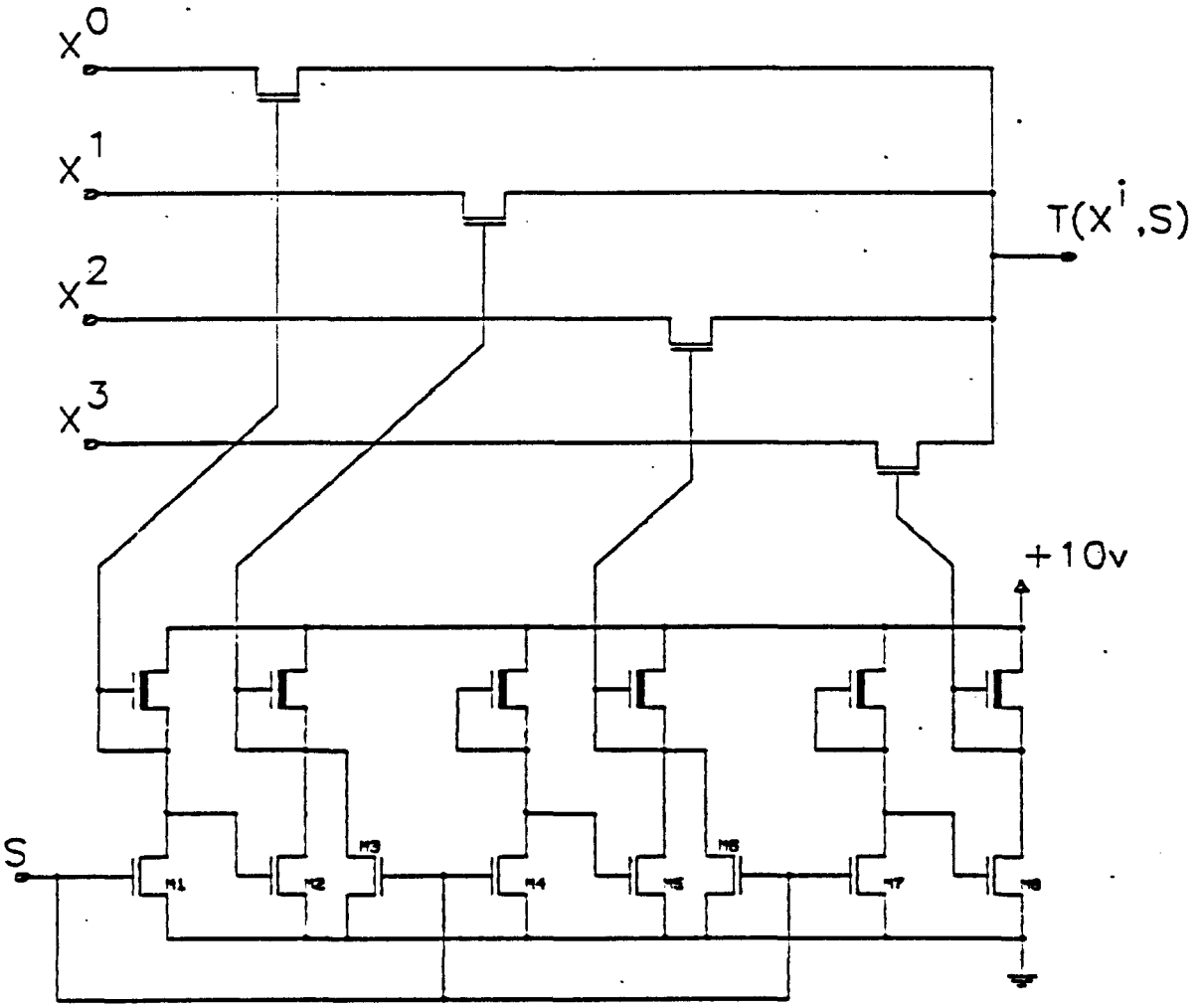
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

transistors with different threshold voltages. In order to determine the data stored at a particular ROM site, one examines the time it takes for a ramp row select signal to turn on the transistor. As a result of this design approach the access time of the ROM is considerably slower than that of the previous ROM or that of a corresponding binary design. This did not render the idea useless. In fact, this technique has been used in several devices that were in commercial production at the time the original article was written. The devices were targeted for systems that did not require extremely fast operation so that the slow access time was not a problem. By being able to use multi-valued logic for the ROM, the designers were able to reduce the area occupied which of course lead to reduced costs and improved yield.

There has been some logic design work done using a quaternary system. In [KaHi84], the authors describe an image processor whose basic element was the quaternary t-gate. As shown in Fig. 7, an NMOS process capable of producing depletion mode transistors and varying thresholds is required. The authors report, as a result of Spice simulations, a delay time through the t-gate of around 100ns.

An example of a voltage mode circuit based on a bipolar design was presented in [ShAb82]. In this paper, a 16x3 quit ROM was used as part of a digital waveform synthesis circuit. At each memory location, the data was programmed by connecting the collector of the transistor to a fixed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Threshold Voltages							
M1	M2	M3	M4	M5	M6	M7	M8
0v	2v	2v	2v	2v	4v	4v	2v

Depletion Mode Transistors: −3v
Pass Transistors: 1v

Figure 7. Quaternary T-Gate

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

reference voltage. The four logic levels 0 through 3 corresponded to the equivalent voltages, 0 through 3 volts. The select lines for the ROM were derived from binary inputs. The output of the ROM was fed into a QAC, quaternary to analog converter, so there was no need to perform any logic functions on a quaternary signal. No discussion of the speed or size of the ROM was given.

Make and Break operators are the basis for the design work presented in [ZuAf85]. The make and break operators are the equivalent of switches that close and open respectively when the input control variable exceeds a certain threshold level. Specifically, then, for the make operator:

$$M_i(c) = \text{short circuit if } c \geq i$$
$$= \text{open circuit if } c < i$$

and for the break operator:

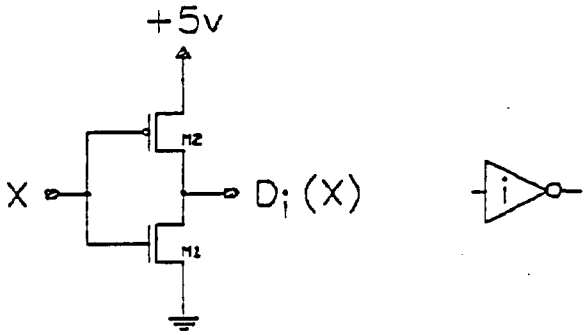
$$B_i(c) = \text{open circuit if } c \geq i$$
$$= \text{short circuit if } c < i.$$

Before the circuit that implements these operators can be constructed, the literals, $D_i(X)$ must be generated, see Fig. 8a. These literals are defined as:

$$D_i(X) = \begin{cases} 3 & \text{if } 0 \leq X \leq i \\ 0 & \text{if } i < X < 4. \end{cases}$$

With these literals available, it is quite easy to implement the make and break operators, Fig. 8b. In the article the authors demonstrate a design methodology using these operators. Results of their Spice simulations show that the maximum delay of the literal circuits is about 35ns.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Vd = 5/3volts

Threshold Voltages		
i	M1	M2
0	0.5Vd	2.5Vd
1	1.5Vd	1.5Vd
2	2.5Vd	0.5Vd

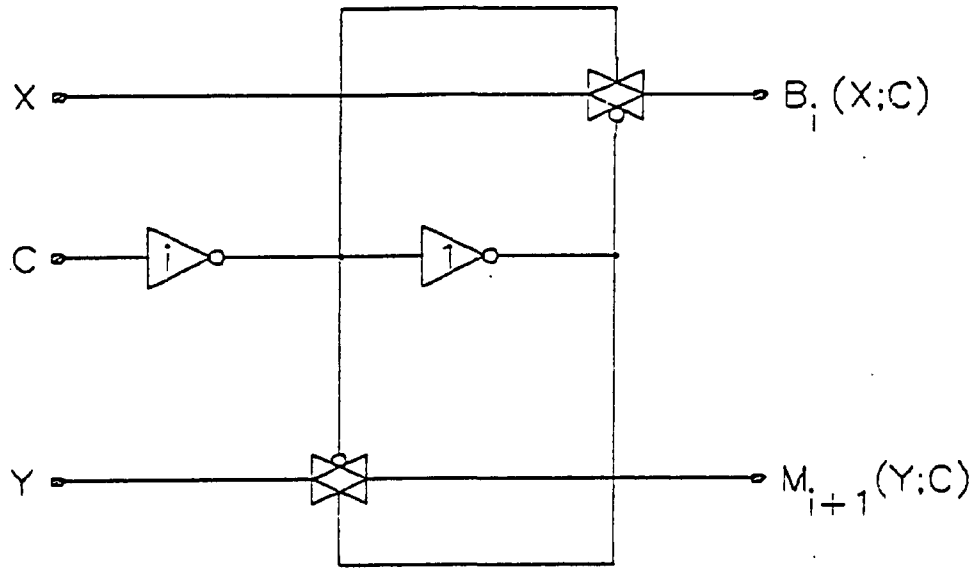


Figure 8. Make and Break Operators

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

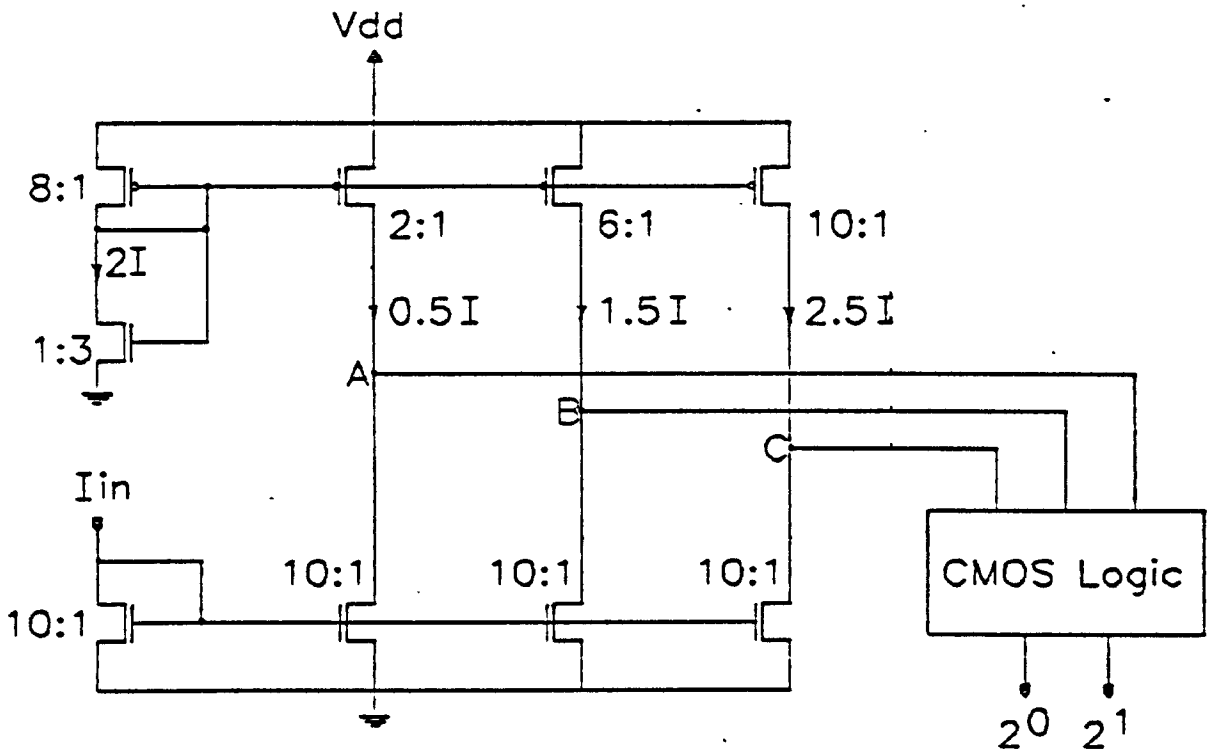
2. Current mode logic

The principle behind current mode logic is to represent the various logic levels as integer multiples of a unit current. One of the advantages claimed by this type of logic is the ability to perform operations such as addition directly through the analog summing of currents. One drawback, however, is the difficulty with which any given signal is routed to more than one destination.

A set of circuits for encoding and decoding quaternary signals was presented in [FrCu83], refer to Fig. 9. The encoder takes two bits of input and converts them to a single current signal as an output. In contrast, the decoder takes a current input and generates the corresponding bits. With these two circuits it is possible to go from a CMOS binary logic system to a quaternary current mode system and back again. No circuitry was discussed for operating on the quaternary signals themselves.

The Quaternary Threshold Logic Full Adder has been under investigation for some time. In [CuWh80] and [WhCu83] the authors implemented the adder using bipolar technology. In [CuFr85], the authors developed the same circuit using a MOS process, see Fig. 10. The two input currents are fed into the I_{in} node where they are summed. The remainder of the circuitry is used to generate the carry signal and to correct the sum if a carry was generated. The transistors

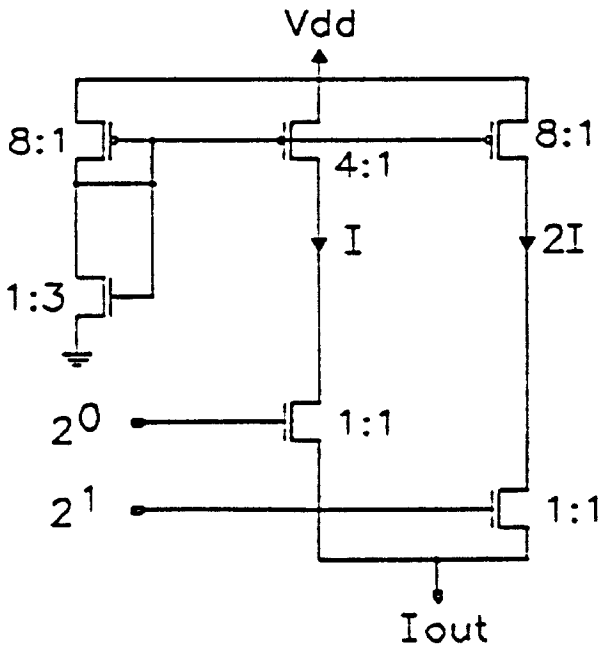
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Quaternary to Binary Decoder

$$2^0 = \overline{B}$$

$$2^1 = \overline{C\overline{B}+A}$$



Binary to Quaternary Encoder

Figure 9. Quaternary Encoder and Decoder

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

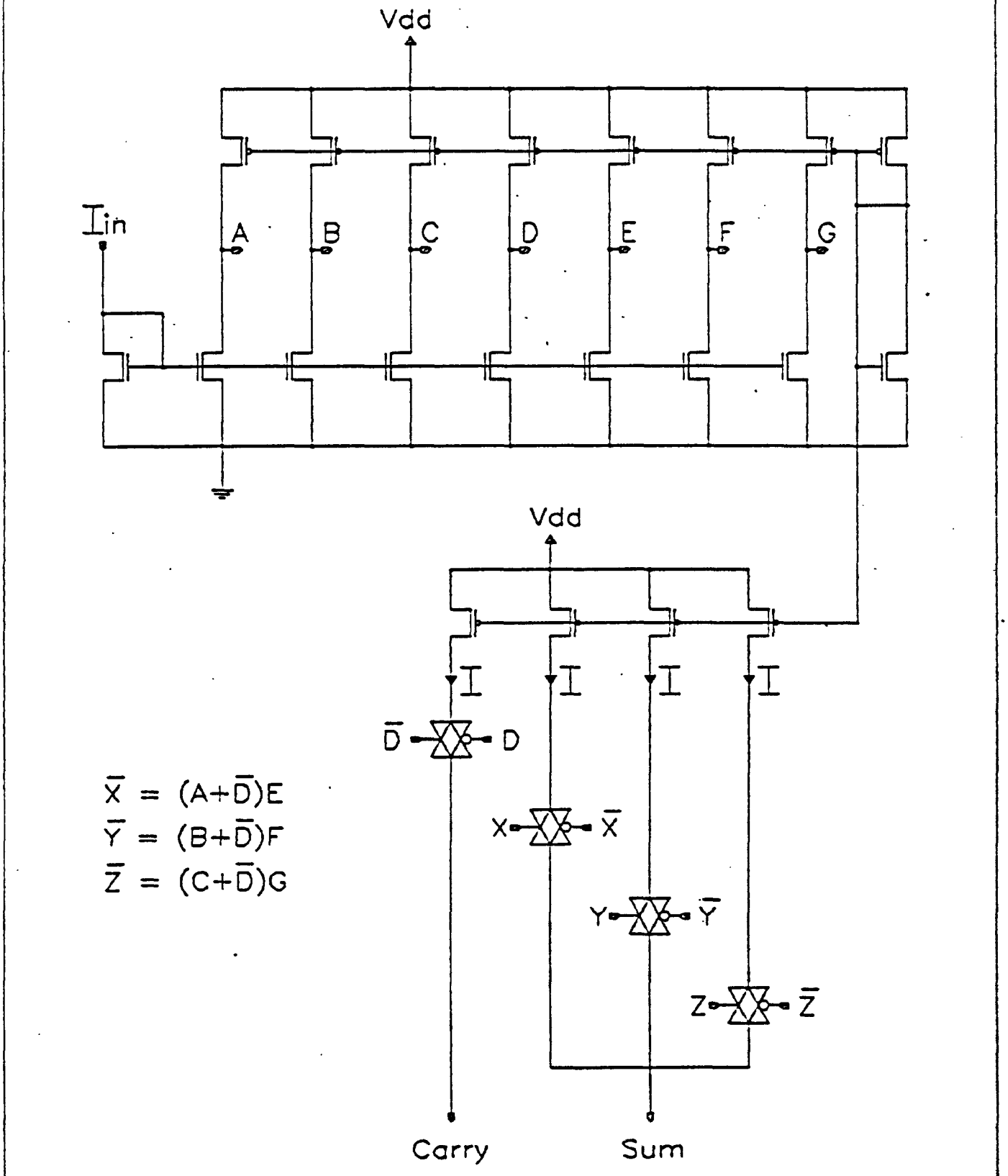


Figure 10. Quaternary Logic Full Adder

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

M2 and M9 through M8 and M15 form seven current comparators. The outputs from these comparators are sent into a logic network which then controls the transmission gates. The sum signal is in the range of 0 to $3I_0$ and the carry is either 0 or I_0. This circuitry is geared towards a specific problem and is not applicable to general multi-valued logic design.

A fairly detailed multi-valued design effort was conducted in [SoEs86]. In this paper, the authors present the complete design of a Residue Number System FIR digital filter using multi-valued logic. The circuitry and simulations are based on an ECL process. The operation of this design is more complicated than the circuits that have been examined thus far and for this reason, a complete analysis will not be carried out. Interested readers should consult the article itself for more details. A discussion of the results from this article will be deferred until the end of this report.

One quick point that can be brought out from this paper is the fact that the design was based on circuits using eight logic levels. This illustrates another advantage that current mode logic has over voltage mode logic. Since the various logic levels are represented by current flow, the number of logic levels and the noise margins are virtually independent of one another. In contrast, for voltage mode logic, there is a direct relationship between the number of logic levels and the noise margin. For the latter case, given a fixed voltage supply, the separation between logic levels decreases as more logic levels are added consequently

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the noise margin will increase.

B. Selection of a Multi-Valued Technique

Part of the research for this thesis required the design of a multi-valued circuit so that results could be obtained first hand. This, in combination with the work done by others, would provide the information upon which the conclusions of the thesis were based.

Before the design work could begin, a suitable multi-leveled technique had to be adopted. Whatever technique that was ultimately selected, it had to match the fabrication process that was available to the University. As it turns out, this is a CMOS 3 micron double metal process developed by Northern Telecom. In this process one can create nMOS and pMOS enhancement mode transistors only. Furthermore, each transistor type has only one fixed threshold voltage which is not under the designer's control. The process has evolved from 5 micron single metal to 3 micron single metal and finally to its present 3 micron double metal form. This, of course, means that techniques involving bipolar transistors could not be used. In addition, of the MOS designs, those requiring depletion mode transistors or a number of different threshold voltages, were also ruled out. Of the remaining designs, the one proposed by Mouftah [MoGa84] looked the most promising.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

II. Detailed Analysis of the Ternary System

From the discussion in the preceding section, it is clear that of the multi-leveled techniques described, the one proposed by Mouftah is the most suitable for our fabrication process. Before using the technique as presented, however, some time was spent investigating possible variations and/or improvements that might be made. Unfortunately, the work in this direction was not terribly fruitful. The designs either had longer delay times or required more area than those of Mouftah.

There were several key parameters of the pti and nti, though, that were modified. Mouftah's original transistor sizing was geared to the process characteristics of the 5 micron process. There are a number of differences between this process and the 3 micron one that the current ternary design was to be fabricated with. As a result, the widths and lengths given in Mouftah's work were changed. With the original transistor sizing, Mouftah increased the length of the gate channel in order to modify the impedances of the design. In the current designs, the widths of the transistors were increased instead. These two approaches represent a trade off between circuit speed and power dissipation. Even though the static transfer curves of the original and current ternary inverters may look the same, the original ones will consume less power yet have longer

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

delays. The reasoning behind this is quite straight forward. If one compares the symmetrical CMOS inverter characteristic with that of the pti, see Fig. 11, one sees that the curve is shifted to the right. This results either from increasing the strength of the pMOS transistor, by increasing it's gate width, or weakening the nMOS transistor, by increasing it's gate length. Thus there are two approaches to achieving the same result as far as the static characteristics are concerned. From the point of view of circuit speed, however, the design using weaker transistors will have less drive capability and consequently, slower operation.

For this ternary system, using the notation described earlier, we have the following:

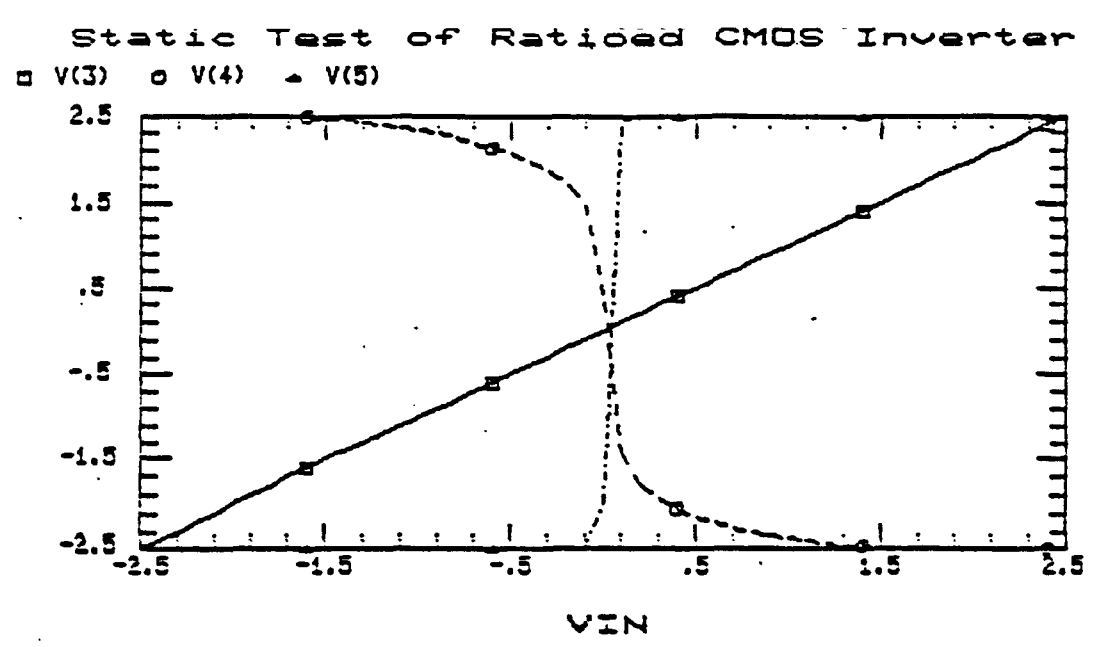
$$S = \langle 0,1,2 \rangle$$
$$R = \langle -2.5,0,2.5 \rangle$$
$$T = \langle -1.0,0.5 \rangle$$

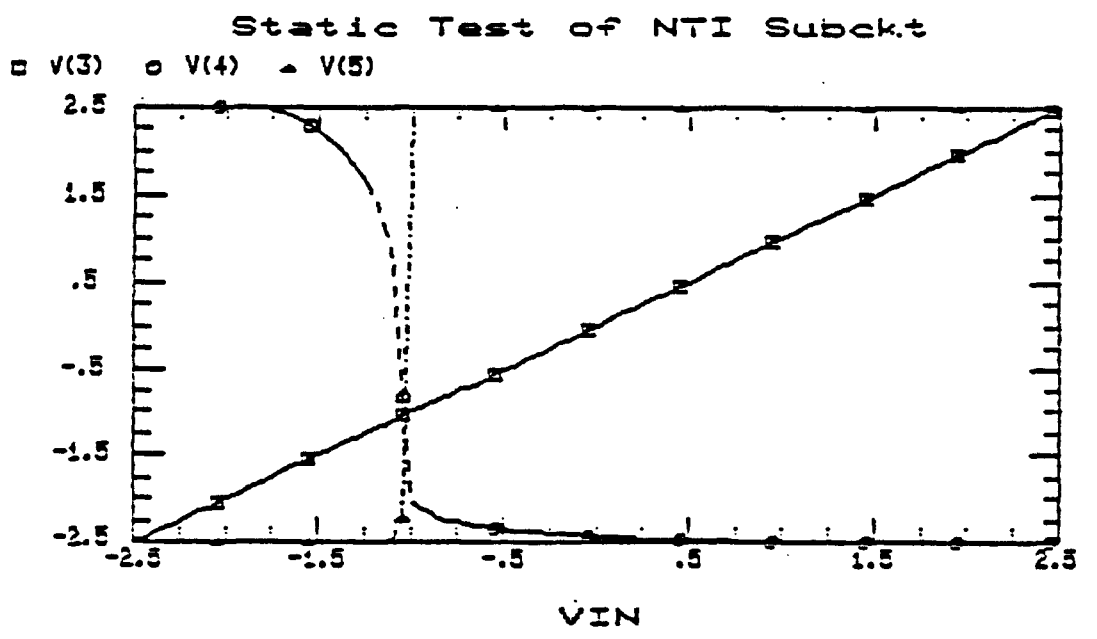
and, repeating here for the sake of completeness, the operation of the two literals that are being implemented:

$$\text{PTI: } U'\langle 0,1,2 \rangle = \langle 2,2,0 \rangle$$
$$\text{NTI: } U'\langle 0,1,2 \rangle = \langle 2,0,0 \rangle .$$

The proper widths and lengths of the devices were fine tuned and the operation of the circuits verified using SPICE, (Simulation Program with Integrated Circuit Emphasis). The results of the static tests of the two ternary inverters are shown in Fig. 11. The circuits were also tested to determine the static power dissipation. The static dissipation is more important than it is in normal CMOS circuitry since, when the input is a logic 1, both nMOS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.





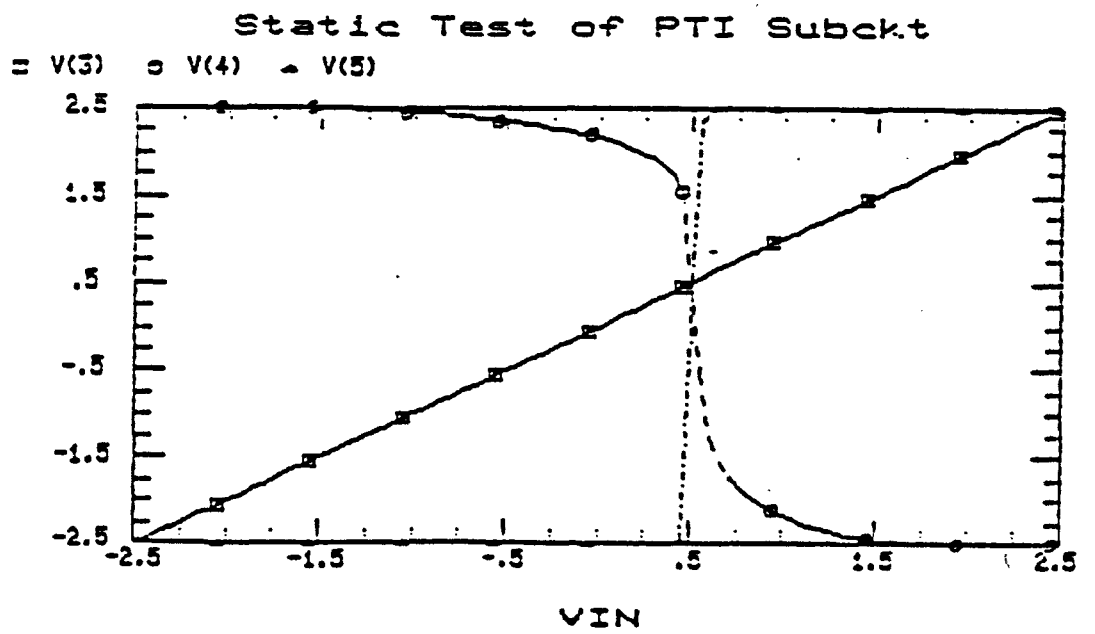


Figure 11. DC Transfer Characteristics of Several Inverters

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and nMOS devices are designed to be on at the same time. In a standard CMOS inverter, both devices are only on during the switching from one logic level to the other. The graphs of the supply current for the nti and pti are shown in Fig. 12. For the logic 1 input, 0 volts, the currents for the nti and pti are 70 uA and 120 uA respectively. From this, knowing that the power supply is 5.0 volts, we can calculate the power dissipation to be:

NTI: 0.35 mwatts
PTI: 0.60 mwatts

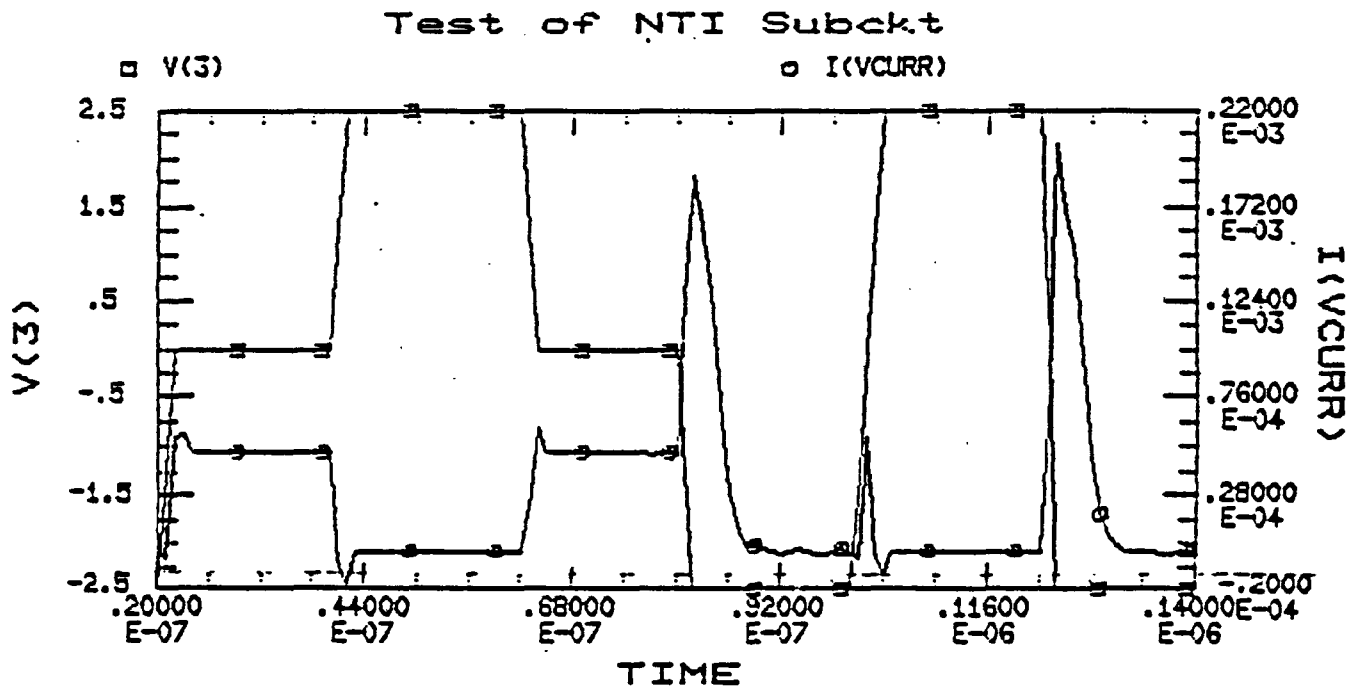
Now if we assume that the input is a logic 1 for only one third of the time, then the average power dissipated becomes:

NTI: 0.11 mwatts
PTI: 0.20 mwatts

We are also interested in the switching speed of these circuits. The switching speed is directly related to the load placed on the output of the gates. In order to get a delay time that represents the value one would expect in a real circuit, the output was loaded with a similar circuit. Also, in an effort to simulate the finite rise time of the real world signals, the input voltage supply used to test the circuits was given specific non zero rise and fall times. The results of these dynamic tests are shown graphically in Fig 13.

The other two remaining circuits that make up this ternary system, other than the ROM, are the decoders and encoders. The function of the decoder is to take a ternary input signal and generate three mutually exclusive binary

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



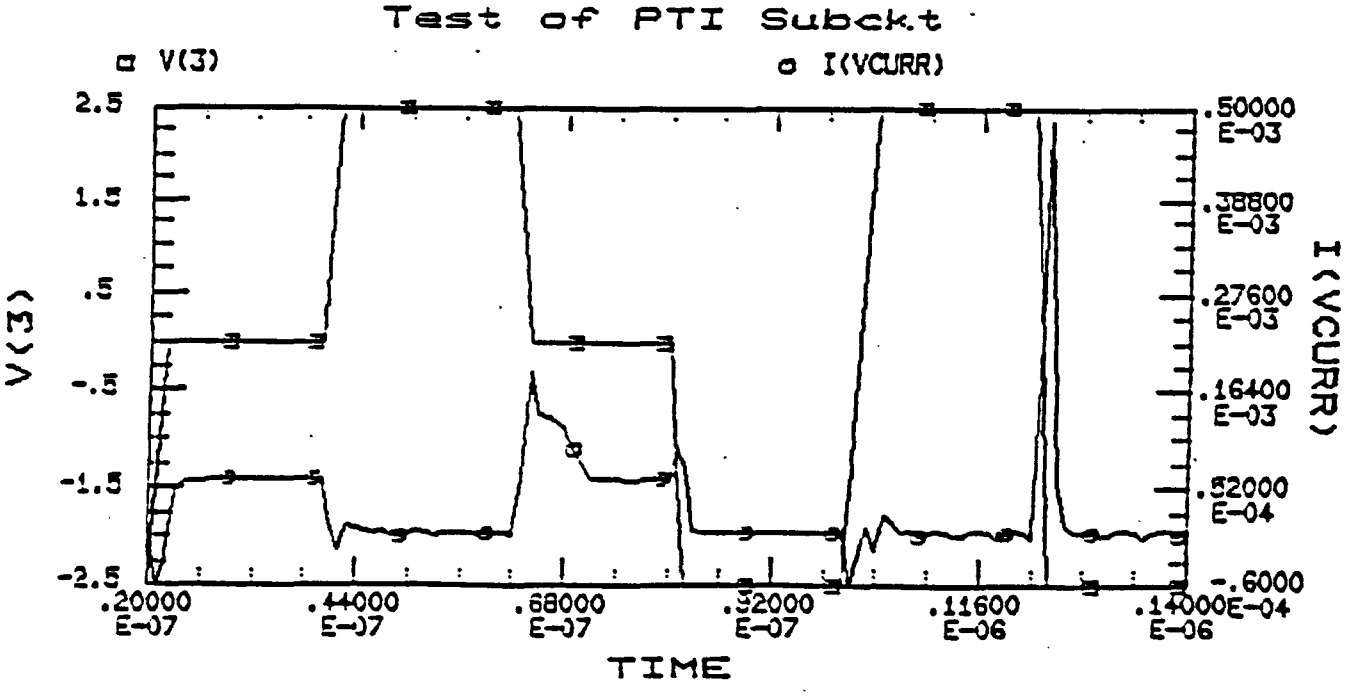
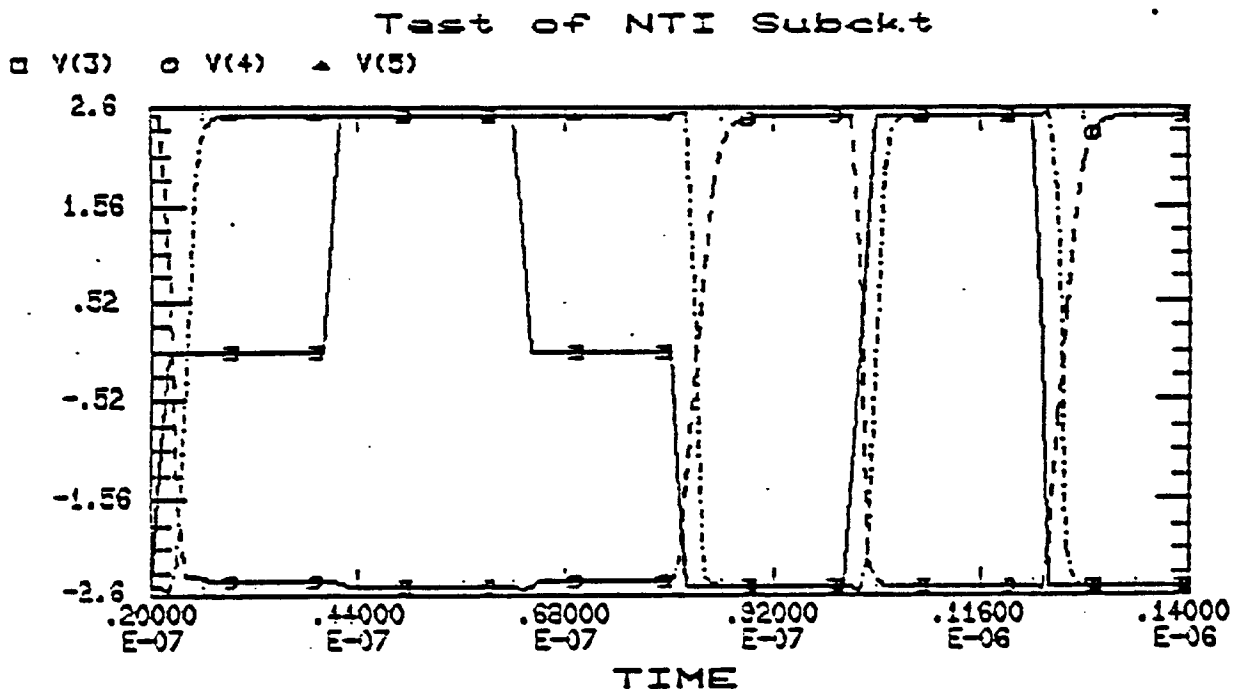


Figure 12. NTI and PTI Supply Currents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



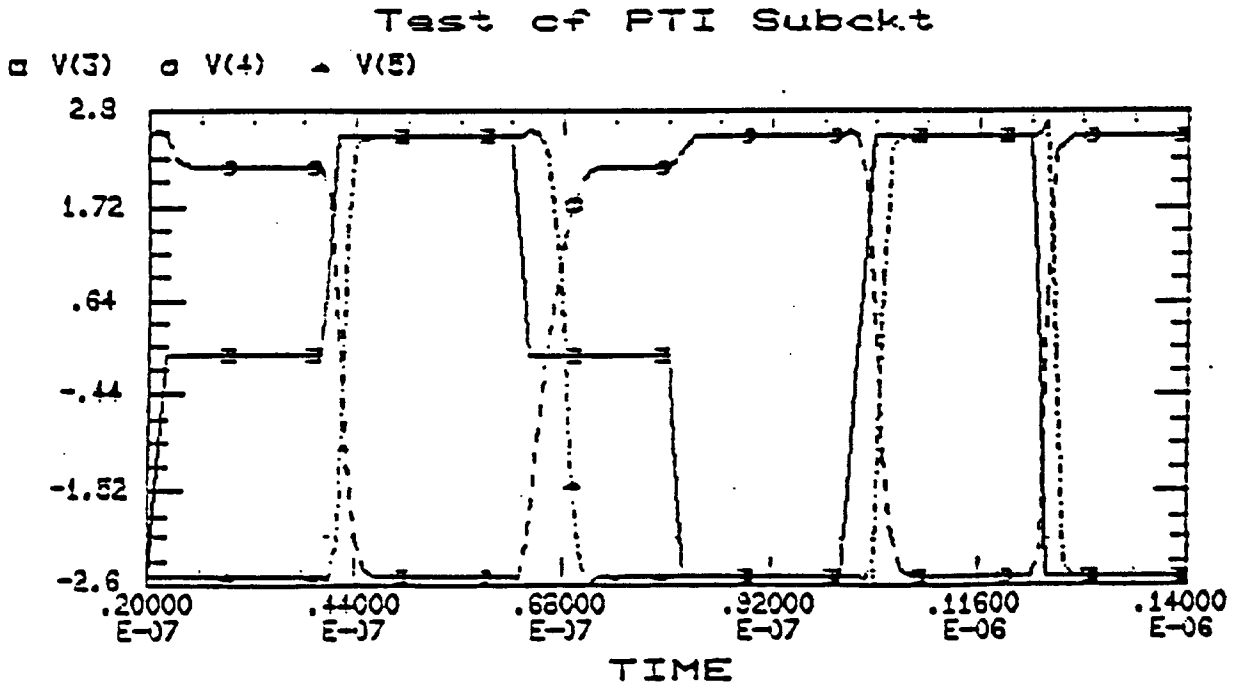
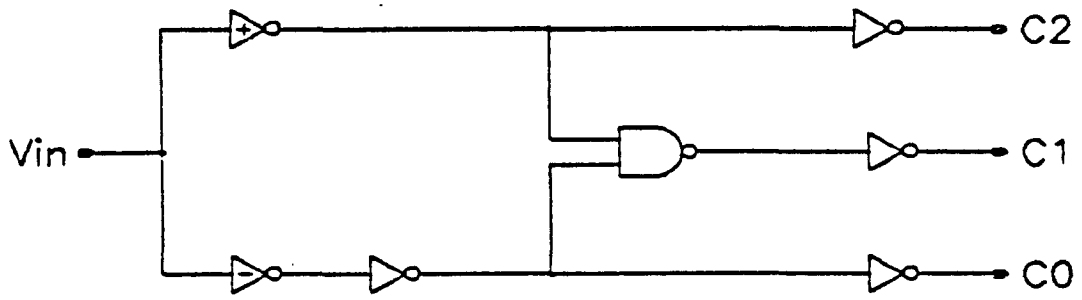


Figure 13. Dynamic Tests of the NTI and PTI

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

output signals. Each signal corresponds to a particular logic level and is only high if the input has that logic level. These signals are produced by a small combinatorial network which has, as its inputs, the outputs from the nti and pti. The circuit diagram for the decoder is shown in Fig. 14. The encoder does just the opposite, taking three mutually exclusive inputs and generating a ternary output. Note that the inputs can be fed directly from the outputs of the decoder, with the exception of one signal that needs inverting. This circuit can also be found in Fig. 14. No encoder circuit was developed as part of Mouftah's design since he relied almost entirely on the t-gate. The simulation results of the encoders and decoders along with the ternary inverters can be found in Appendix III.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Vin	C2	C1	C0
0	0	0	2
1	0	2	0
2	2	0	0

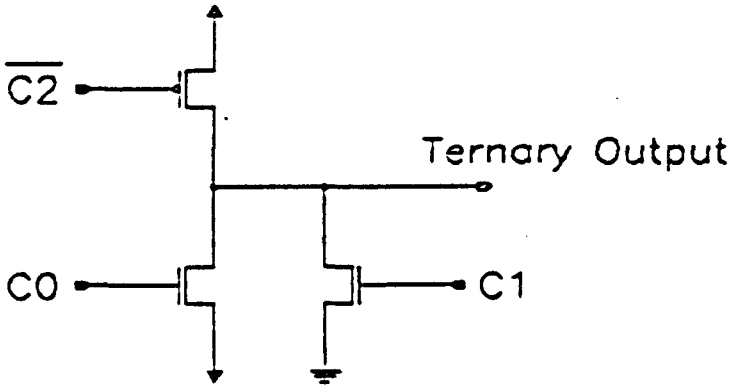


Figure 14. Encoders and Decoders

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

III. RNS Addition, a Representative Design Problem

As was stated in the introduction, besides doing a survey of the work already done by others, it was desirable to do some independent design work and see if the results obtained from this work corroborated that obtained from the survey. It was also felt that the design problem chosen should be representative of the type of designs that have arisen, or will arise, as a result of the research that is being conducted by some members of the Signals and Systems Group at the University of Windsor.

The problem that was selected was that of RNS addition. Before delving into the details of the RNS addition it would be appropriate to review some of the basic ideas behind the residue number system.

If we have some number, X, in the decimal system and we wish to express it in its RNS form we get:

$$X = (x_0, x_1, \ldots, x_{N-1}) \tag{2}$$

where $x_i = X \bmod m_i$ = remainder of X/m_i

The x_i are the residues of the moduli, m_i. If we choose the moduli such that they are all relatively prime, ie. no common factors, then the decimal equivalent of the number is constrained by:

$$0 \leq X < M, \tag{3}$$

$$M = \prod_{i=0}^{N-1} m_i \tag{4}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Each number, X, provided it is bounded as above, has a unique representation in terms of its residues with respect to the moduli. Given such a number in RNS form, it can be converted to decimal form through techniques such as the Chinese Remainder Theorem. We will not go through the details of this technique but instead refer the reader to [Gard59] which also provides a good introduction to the RNS. The operations of addition and multiplication can be performed on the residues independently of one another. This property is one of the main reasons for the interest in the residue number system. Specifically then, we may write:

$$X \circ Y = (x_0 \circ y_0, x_1 \circ y_1, \ldots, x_{N-1} \circ y_{N-1}) \qquad (5)$$

where $x_i \circ y_i$ = modular addition or modular multiplication with respect to the modulus, m_i.

The significance of this is that each residue requires fewer bits to be represented than does the original number. The the operations of addition and multiplication operate on smaller numbers and can thus be computed in less time than it would take to operate on the original number. Furthermore, since the operations on the different residues can be computed in parallel, it is possible to obtain the results of these operations in the same amount of time regardless of the number of moduli used. This, of course, means that the resolution of the system, determined by the number of moduli used, does not affect the time required to perform addition or multiplication.

Let us take a quick example to illustrate these points.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Suppose we choose as our set of moduli the numbers; 32, 31, 29 and 27. Now from equation 4 we know that the largest number that we can represent in this system is:

$$M = 32 \times 31 \times 29 \times 27 = 776736$$
$$\simeq 2^{19.57}.$$

Thus to represent the same number in binary would require 20 bits whereas the residue with respect to each modulus can be represented in only 5 bits since the residues take the values 0 to 31. Continuing, let us add the numbers 1000 and 462. In terms of their residues, using the form of equation 2, these numbers are:

	1000 =	(8, 8,14,1)	
	462 =	(14,28,27,3)	
	----	----------	
adding gives:	1462	22,36,41,4	(a)
		(22, 5,12,4)	(b)

where line (b) comes from the remainder of line (a) with respect to the corresponding moduli.

Now if we double check the residue representation of 1462 we find:

$$1462 = (22,5,12,4)$$

which is what was expected. If this operation were performed in hardware, it would be possible to have four 5 bit modular adders operating in parallel. The speed of the circuit would be governed by the time delay through one 5 bit adder as opposed to the delay through a 20 bit adder. Of course this does not take into account the time needed for encoding the inputs into rns form and that needed for decoding the result into decimal form. The use of an RNS based system really only becomes advantageous if one has a large number

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of addition and/or multiplication operations to be done and one only needs to encode at the beginning and decode at the end of these operations.

One possible approach to modular addition has been to use the two values to be added, as an address into a ROM [SzTa67]. The contents of the ROM then represent the modular addition of the two inputs. A variation of this approach put forth in [Tahe87] introduces pipelining into the design, thereby increasing the throughput rate of the system. The technique for 5 bit modular addition becomes quite clear if one examines the relevant equations. To add two 5 bit numbers, A and B, we have:

$$(A + B) \bmod m = (2^4A_4 + 2^3A_3 + 2^2A_2 + 2^1A_1 + 2^0A_0 + B) \bmod m$$

Now if we take a 5 stage approach to the same equation:

$$\begin{aligned} s1 &= (2^0A_0 + B) \bmod m \\ s2 &= (2^1A_1 + s1) \bmod m \\ s3 &= (2^2A_2 + s2) \bmod m \\ s4 &= (2^3A_3 + s3) \bmod m \\ (A + B) \bmod m &= (2^4A_4 + s4) \bmod m \end{aligned}$$

Each stage has a 5 bit input (B,s1 to s4), which represents the address of memory location inside a small ROM, and a selection variable, A_i. Referring to Fig. 15 now, if the A_i input is 0 then the output of that stage equals the input thus the selection variable directs the input around the ROM by closing switch sw1 and opening sw2. On the other hand, if A_i is 1, then the output equals the input plus the appropriate power of two, and the selection variable sends

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

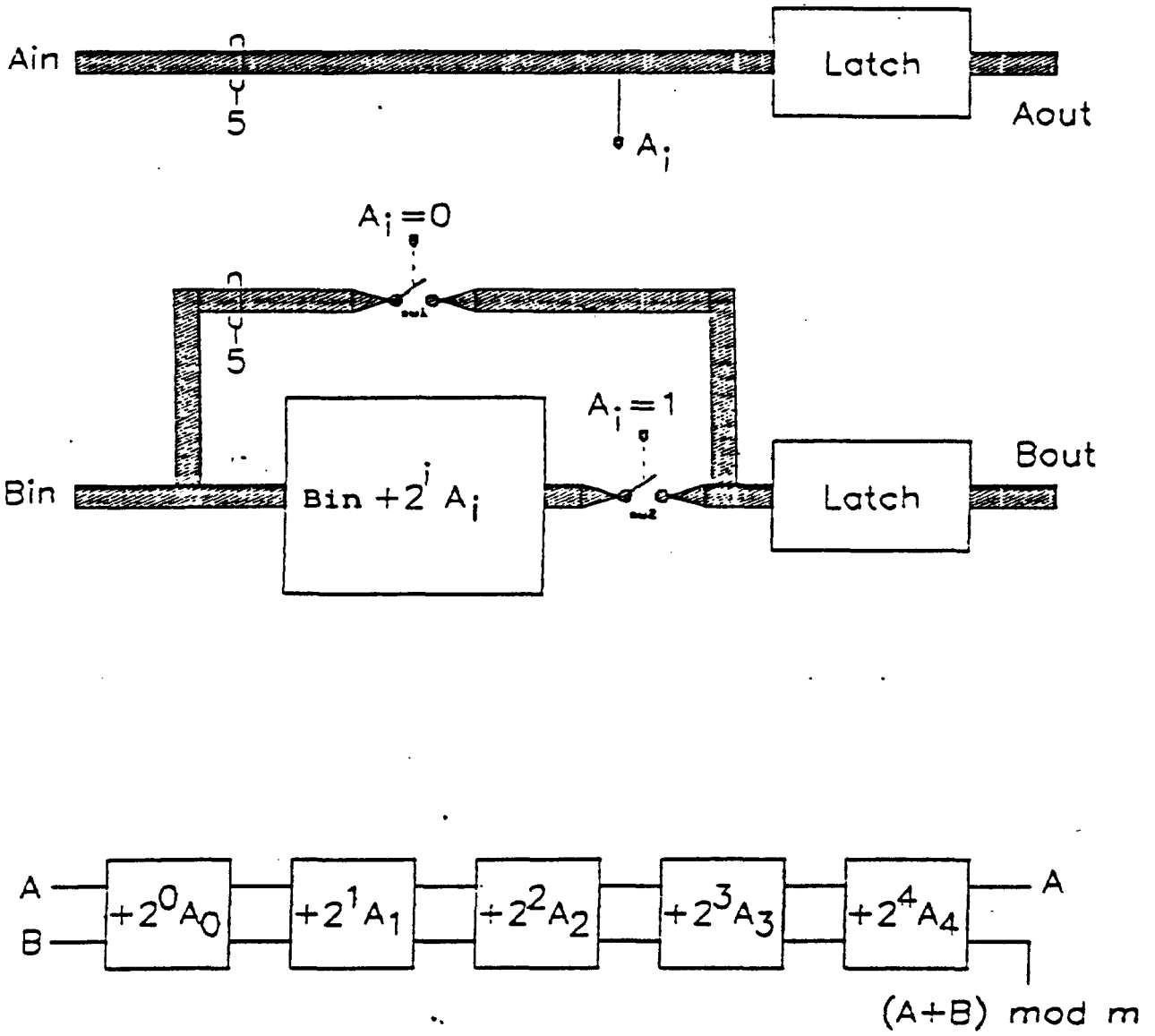


Figure 15. Modular 5 Bit RNS Addition

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the input through the ROM by closing sw2 and opening sw1. Cascading five of these stages, with the ROMs programmed with the proper data, will produce the desired output of A + B mod m.

Pipelining is achieved by including latches in between each stage. Thus, after the first cycle, the output from the first stage is stored in the latches at the end of the first stage. Now it is possible to load another set of A and B values into the first stage. During the second cycle, the first set of A and B are being processed in the second stage while the second set of A and B are being processed in the first stage. This continues until the end of the fifth cycle at which time the sum of the first A and B set is available at the output of the circuit. There are also four other sets of A and B values, each at a different stage in the pipeline. From this discussion one can see that it takes five cycles before the first sum is available, this is called the latency time of the circuit. Once the pipeline is full, ie. after five cycles, then a new sum is available at the end of each cycle.

This pipelined structure, besides being used for RNS addition, can been applied to a number of other DSP design problems. By making the appropriate modifications, it is possible to use the same basic structure for things such as an FFT butterfly, RNS multiplication and conversion from binary to RNS [TaJM87a]. It has also been shown that circuitry for a digital FIR filter can be implemented using hardware based on this RNS adder structure [TaJM87b].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The application of this module to a broad range of designs made it ideal for the study required by this thesis. It was felt that the results obtained from a comparison between a multi-valued and binary implementation of this design would give a good indication of the relative merits of the two approaches as far as high speed DSP applications are concerned.

Before moving on to the designs, themselves, it would be useful to discuss the differences the ternary system makes to the equations. The development for the ternary system is:

$$(A + B) \bmod m = (3^2A_2 + 3^1A_1 + 3^0A_0 + B) \bmod m$$

In a similar manner, we break this into 3 stages:

$$\begin{aligned} s1 &= (3^0A_0 + B) \bmod m \\ s2 &= (3^1A_1 + s1) \bmod m \\ (A + B) \bmod m &= (3^2A_2 + s2) \bmod m \end{aligned}$$

The first obvious difference is that there are only three stages in the ternary version of this design. There is another more subtle difference that is also important. In the binary case, the selector variable, A_i directed the input either around the ROM or through the ROM. In the ternary case, see Fig. 16, the selector variable can take on three values, 0,1 and 2. This means that we actually need two ROMs. The first ROM adds 3^i to the input and the second ROM adds $2*3^i$. Other than these two differences, the principle remains the same.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

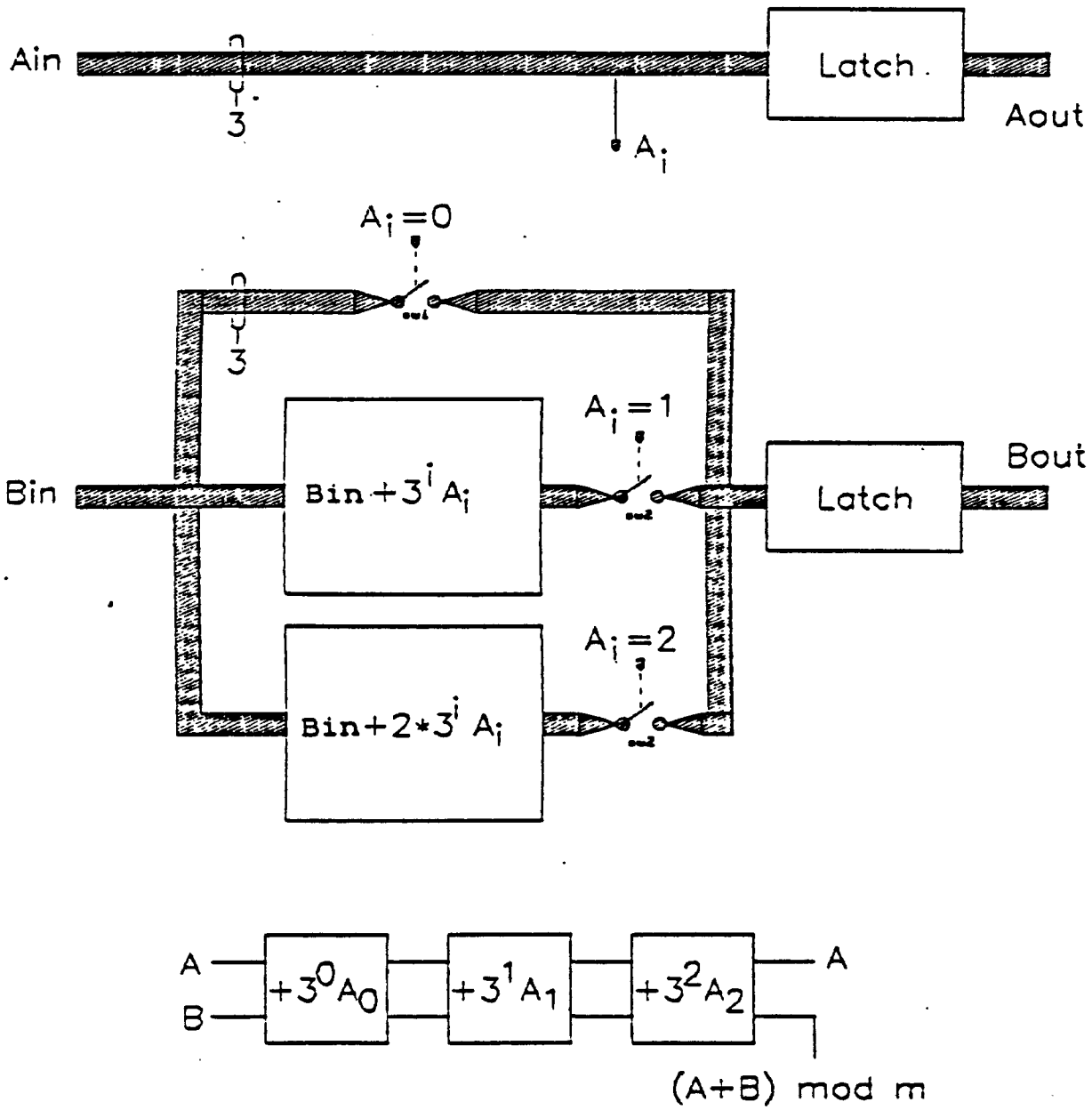


Figure 16. Modular 3 Trit RNS Addition

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

IV. The 5 Bit Binary RNS Adder

The first version of the 5 bit binary rns adder followed the basic structure outlined in the previous section. From this, a second design evolved that incorporated a number of improvements. Both designs will be discussed in detail outlining the architecture, timing and simulations.

A. The First Design

The actual structure of the first design is shown in Fig. 17. The complete schematics can be found in Appendix II. There are two main sections to this design; the latches and the ROM. Both the sections incorporate a mixture of dynamic and static logic. The latches consist of the transmission gates and inverters. The transmission gates connected to the ROM are also used to perform the switching operations shown in Fig. 15.

The latch is designed to function as a master slave flip flop. The latches, Fig. 18a, are controlled by a two phased non-overlapping clock. During the first phase, the input signal is transferred through the first transmission gate and stored on the parasitic capacitance, c1, associated with the gate of the inverter and the routing from the transmission gate to the inverter. Once phase one is over, c1 is isolated from the input. During, phase two, the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

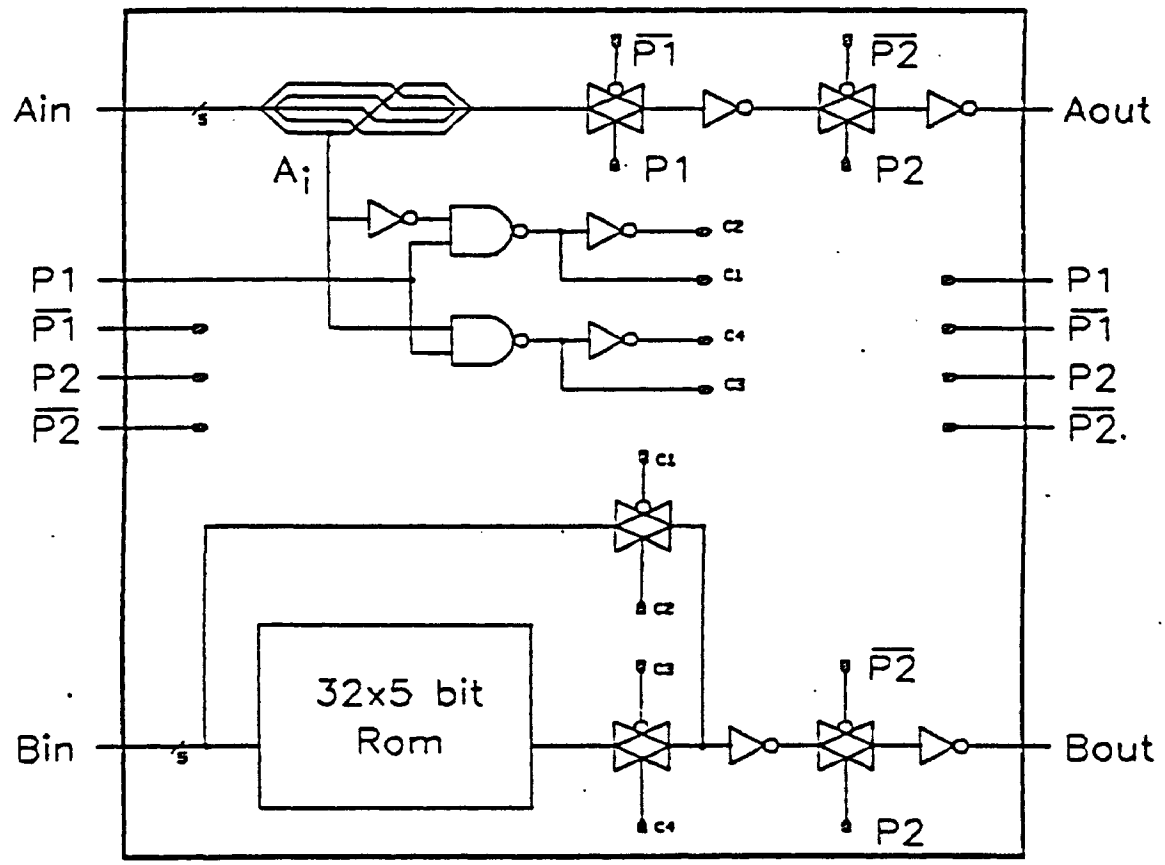
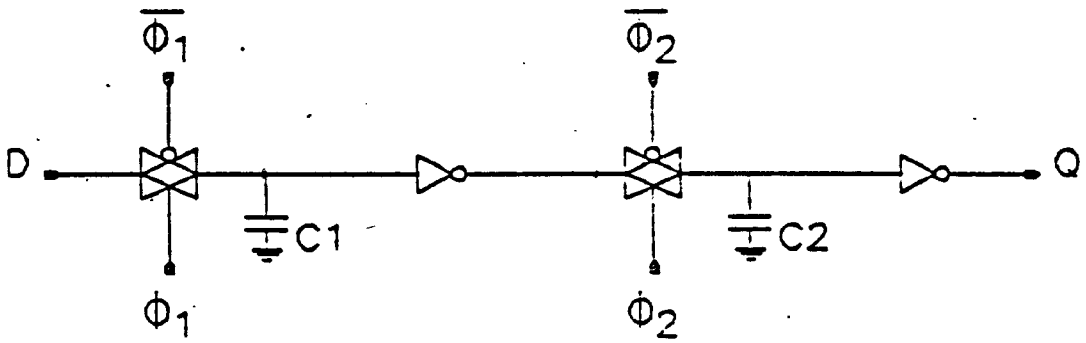
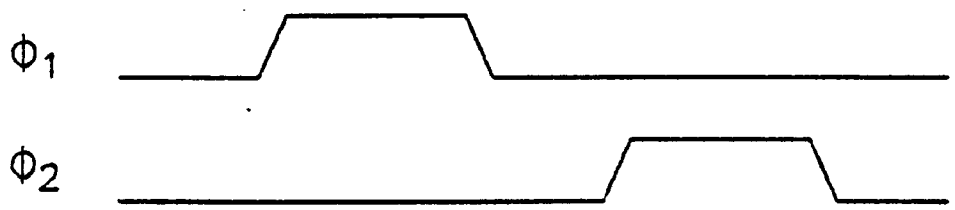


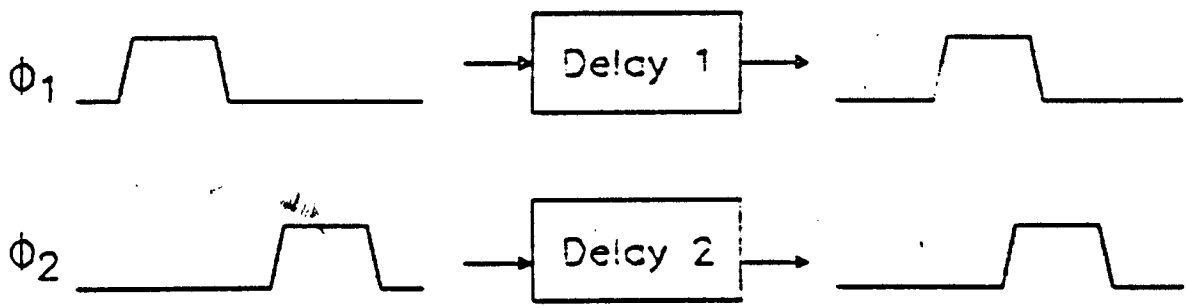
Figure 17. Structure of First Binary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.





a)



b)

Figure 18. Dynamic Latches and Clock Skew

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

second transmission gate samples the output of the first inverter and stores this on another parasitic capacitance, c2. Thus during and after phase two, the correct signal is available at the output of the second inverter.

It is necessary to use a master slave arrangement since the stages in Fig. 17 will be cascaded. We want to be able to have a set of data propagate through the stages in a controlled manner. This brings up another point that must be considered carefully when designing with clock signals and that is the problem of clock skew. In this case, if the two clock signals were to be delayed by different amounts, as shown in Fig. 18b, it might be possible for the two clock signals to overlap. If this occurs then the two transmission gates will be turned on at the same time. When this happens the data can ripple through the next stage and get latched at the output of the next stage, which of course represents an error. It is important that the separation between the two clocks be large enough to accommodate the largest clock skew expected in the design.

The latches for the A inputs are exactly as described above. For the B inputs, the second half of the latch remains the same, however, the first half has to implement the switching operations required by the design. The closure of the switch, sw1 or sw2 from Fig. 15, is accomplished by providing the phase one clock signals to the appropriate transmission gate connected to the ROM. The small combinatorial network above the ROM generates the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

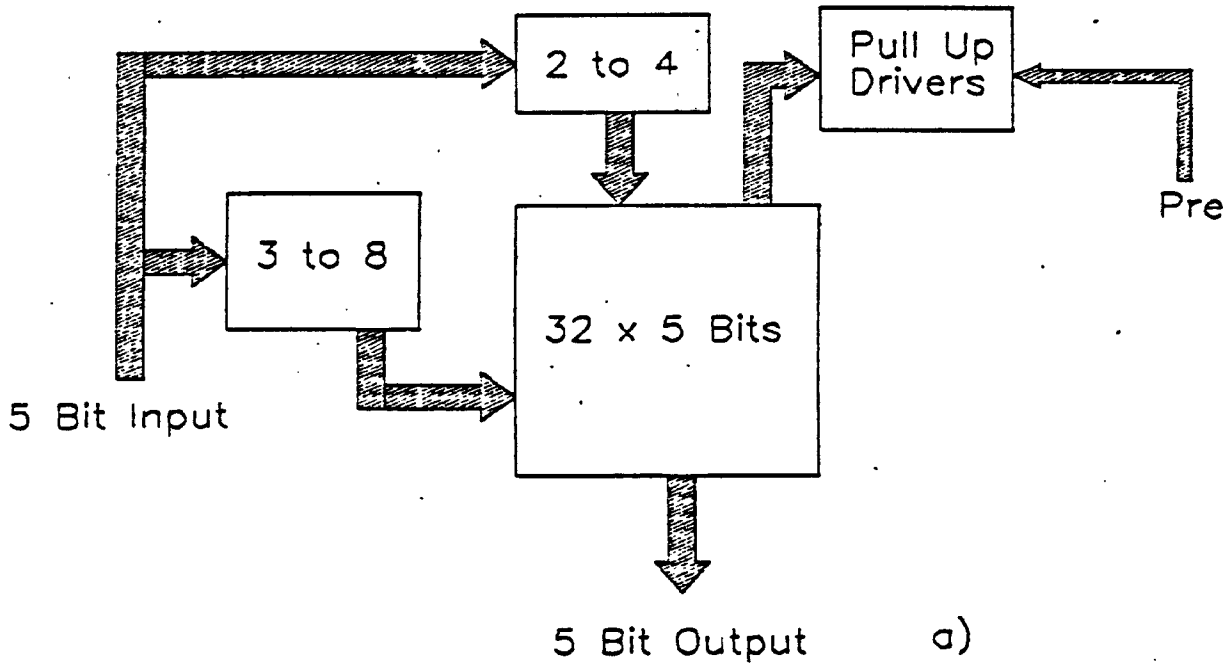
clock signals that are routed to the two transmission gates.

The ROM is configured as 8 rows by 4 columns. This arrangement was chosen so that the width and height of the ROM cell array would be approximately the same. The ROM can be broken down into four functional blocks; the row decoder, column decoder, ROM cell array and pull up drivers, see Fig. 19a.

During the precharge phase of the ROM, which is the same as the phase two clock signal, the row select signals are turned off, ie. all are low, the column select lines are valid, and the pull up drivers are turned on. This charges up the capacitance associated with the bit lines to Vdd, see Fig. 19b. During the evaluate phase the pull up drivers are turned off and the row select lines become valid. If a logic 0 is to be stored at a particular memory location then the column and row select transistors will exist and the bit line will be pulled low. If a logic 1 is required, then neither transistor is produced so there will be nothing to pull the bit line to ground. The ROM is a dynamic structure since the capacitance, which stores the Vdd value for a logic 1, will slowly discharge until the voltage no longer represents a valid logic level. The clock period for the ROM must be smaller than the time it takes for this capacitance to discharge.

The row decoder is the equivalent of a 3 to 8 decoder with one enable signal. The three inputs are the three least significant bits of the B address. The enable signal is fed from the complement of the phase two clock, thus when

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



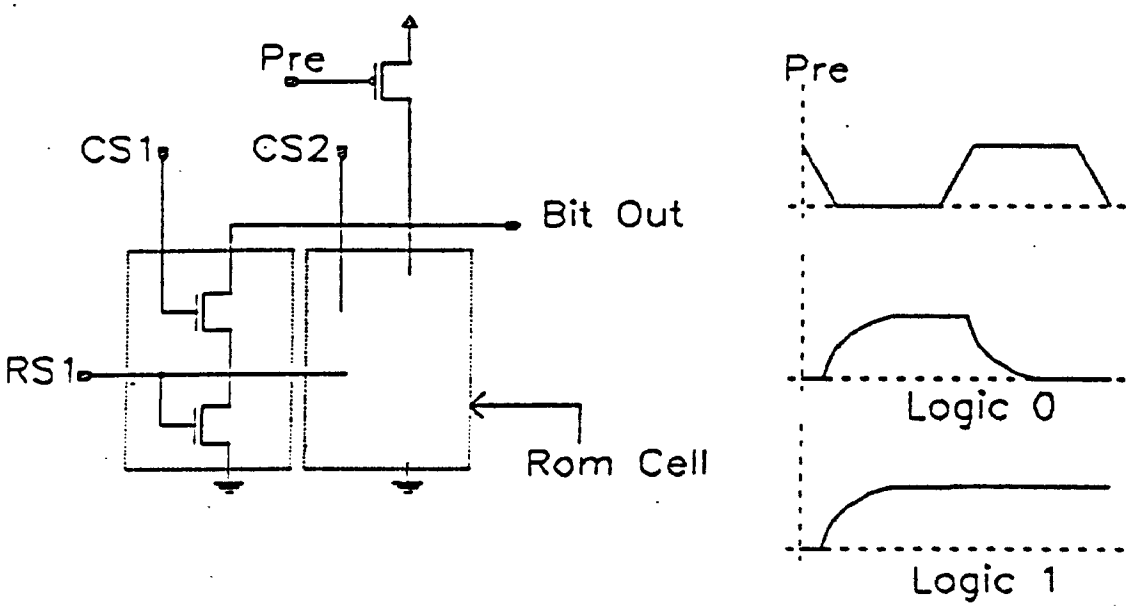


Figure 19. Internal Rom Structure

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

phase two is high and the ROM is being precharged, the row decoder is disabled and all outputs are forced low. As soon as phase two goes low, and the pull up drivers are turned off, the row decoder starts to evaluate.

Similarly, the column decoder takes two inputs, the two most significant bits of the B address, and produces four column select lines. Unlike the row decoder, the column select lines start evaluating as soon as phase two starts. This means that during the precharge phase, the ROM will receive valid column select signals. The reason for doing this is to precharge the nodes between the column select transistors and the row select transistors. This is done to avoid the problem of charge redistribution. Suppose this were not done and all the nodes in the first column had been pulled to ground in the previous cycle. Now we precharge the bit line. Next the precharge phase ends and we turn on the column select transistors in the first column. At this point the charge on the bit line will be shared with the capacitance associated with all the column nodes. This charge sharing can pull down the bit line voltage so that it no longer represents a valid logic level. By charging the column nodes of the column that will be selected, this problem is completely avoided.

The circuit has been carefully simulated using Spice. The layouts of the various sections of the design along with the circuit diagrams and simulation results can be found in Appendix II. From the simulation results, we can construct

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

an accurate timing diagram showing the relationships between the various signals. This diagram is shown in Fig. 20. The power drawn by this circuit is calculated using the following equation:

$$P = CFV^2$$

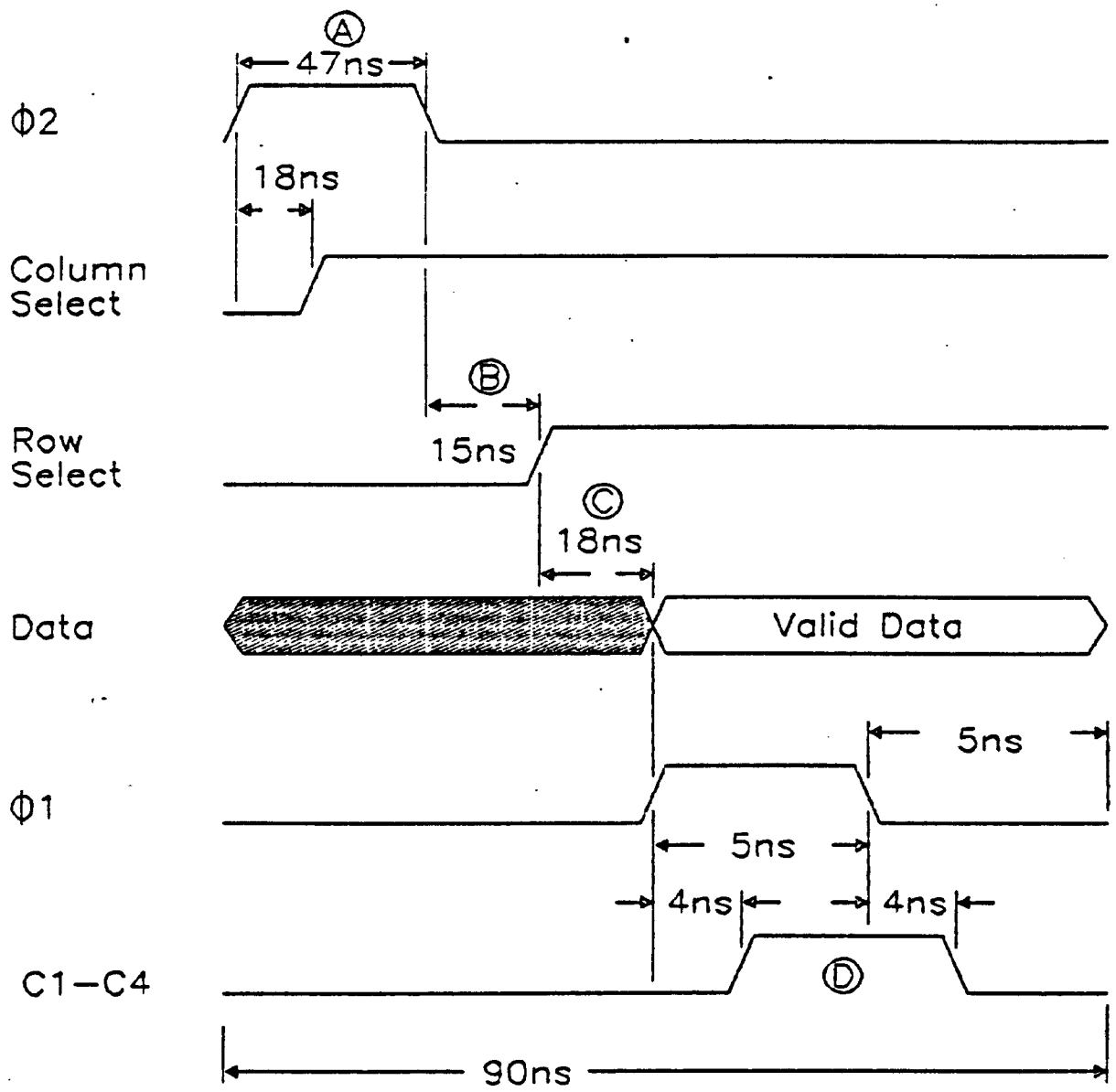
where P = dynamic power dissipated
C = the capacitance of a particular node
V = the voltage to which the capacitance is charged
F = the frequency with which the capacitance is charged and discharged.

This equation is based on the assumption that the static power dissipation of CMOS circuitry operating at high speeds is relatively small compared to the dynamic power dissipation. If a given node is switching between two logic levels, the capacitance associated with that node is being charged and discharged. The current drawn to charge this capacitance gives rise to the dyhamic power dissipation. Clearly, the faster a circuit operates, and the faster the nodes switch, the more power will be used. By calculating the capacitances at the various nodes and making certain assumptions about which nodes were switching it was possible to estimate the power required by the binary design to be approximately 2.2 mwatts per stage. This was assuming the circuit was running at 10Mhz.

B. The Improved Design

There were a number of flaws with the first binary design that were uncovered as a result of extensive simulations of the circuit. In addition, since the improved

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Ⓐ Precharge the bit lines of the rom

Ⓑ Evaluate the second half of the row decoder

Ⓒ Evaluate the rom

Ⓓ Latch the output from the rom

Figure 20. Timing Diagram for First Binary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

binary design was actually done after the ternary design, some of the improvements developed for the ternary design could be incorporated into the new binary design.

The first design flaw that came to light was the fact that it was not necessary to put the column select transistor in each cell. Instead, the column select transistor should be placed at the top of each column. Besides a considerable savings in area, this change also helped to reduce the capacitance on several key signal lines. This meant that the precharge and evaluate times for the rom were reduced.

Another major problem with the first design centered around the row decoders. The last section of these decoders was a standard sized two input nor gate. This gate was supposed to drive the large capacitive load of the row select lines. Instead either the nor gate should have been made stronger or a large inverter should have been inserted between the decoder and the row select line. Without the increased current capability, the rise and fall times of the row select signals were extremely long which contributed to the rather poor performance of the first design.

The final flaw, was the clocking scheme itself. In the the first design, the second half of the row decoder did not start evaluating until after the rom had finished precharging. This was done so that none of the row select transistors would be on at the same time as the pull up driver, minimizing the static power dissipation that would have resulted had they both been on at the same time. If

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

this is changed so that the decoder evaluates completely before the rom precharges, then the time it takes for the rom to produce a valid output can be reduced. Of course, if this were done, there would be static power dissipation, however only two transistors would be on and only during the precharge part of the clock cycle.

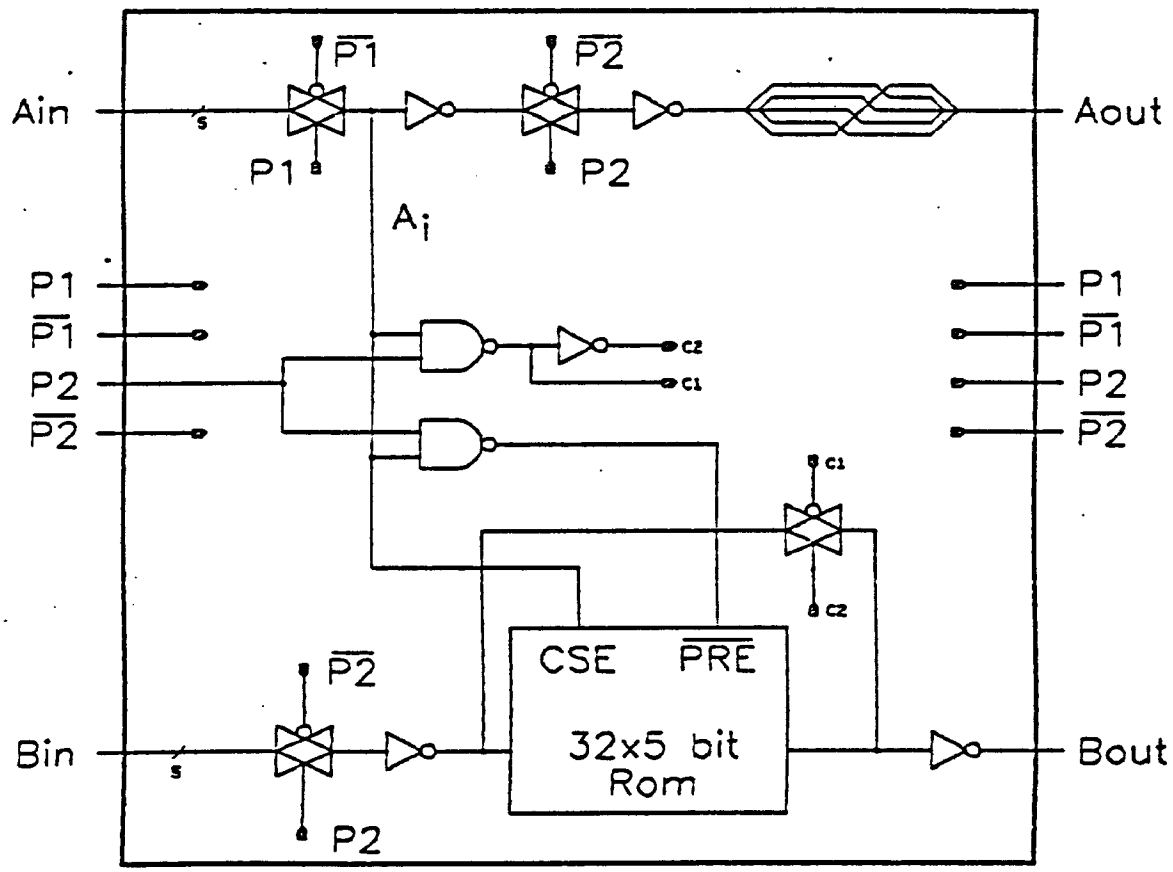
The new binary design was developed to correct the problems just described. The structure of the improved binrary rns adder is shown in Fig. 21a. Complete schematics of this design can be found in Appendix III.

The poor clocking scheme of the preceeding design was corrected, in part, by placing the first half of the latches before the rom. This meant that phase one was used for sampling the output of the previous stage. The circuitry was also designed to allow the row and column decoders to start evaluating as soon as phase one started. Thus the time spent for one half of the latching operation was also used to do the decoding. In the first design, the second half of the latch did nothing but latch the data. This was an inefficient use of the bandwidth of the system.

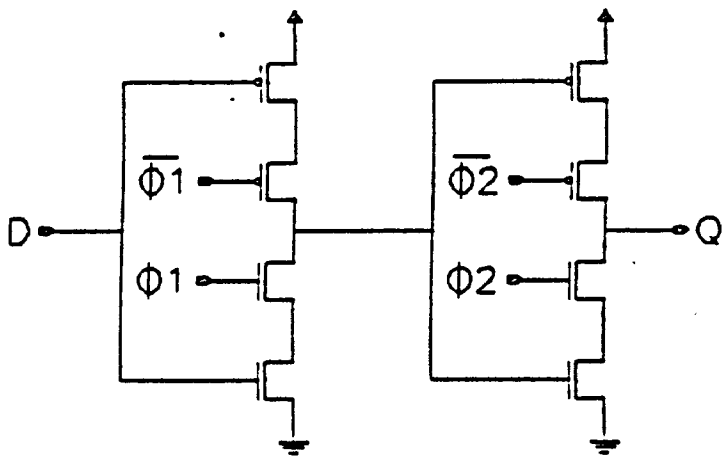
In the first design, the inverters that were part of the latch were there for the sole purpose of being able to drive the next section of the latch. In the new design, the inverters are used in the decoder as well. Using the inverters for two functions helped to eliminate some of the duplication in the circuitry.

Besides making better use of the gates in the latches,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



a)



b)

Figure 21. Structure of Improved Binary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a different type of latch was also used that made more efficient use of silicon. Considering the module for the i^{th} stage, for the B inputs and the A_i input, the transmission gate latches were used as before. For the rest of the A inputs, however, a latch based on the clocked inverter was used. The clocked inverter, Fig. 21b, has a smaller layout compared to the transmission gate latch, unfortunately, it also operates more slowly. In order to avoid the problems of the reduced operating speed of the clocked inverter, they were only used for signals that were driving small loads and that were not in the critical path of the design. The critical path of the circuit took more time than did the clocked inverter that was working in parallel so the fact that the clocked inverter based latch was slower than the transmission gate latch had no effect on the overall speed of the circuit.

The block diagram for the new ROM is the same as that for the first design, see Fig. 19a. The actual transistor arrangement, though, is different and this is shown in Fig. 22. As mentioned before, the obvious advantage of the new design is the reduced area that the ROM array occupies. This change has also led to several other improvements. In the first design, each column select signal had to drive 40 gates, (8 rows x 5 bits), since there was a column select transistor in each ROM site. In the new design the load is only 5 gates, one for each bit, which is an 87.5% reduction. The overall reduction turns out to be only 79% because the routing is longer in the new design. The capacitive load on

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the column decoder, determined by the number of gates it must drive and the length of the routing attached to it's outputs determines how fast the decoder will run. Making this capacitance smaller will result in faster rise and fall times for the column select lines, thereby improving the speed of the ROM. It will also mean a reduction in the power consumed since power is a function of capacitance.

For the row select lines, the reduction is minimal since the same number of gates have to be driven. Still, it was possible to reduce the width of the ROM since it was no longer necessary to route the column select signals through the ROM sites. Having a narrower ROM meant that the interconnecting poly from column to column is shorter. To be specific, the current row select capacitance is around 98% of the old capacitance.

Looking at the bit lines in the first design, one sees that the drains of all the column select transistors, for a particular bit, were connected together. This led to a rather large bit line capacitance and a rather slow precharge and evaluate time for the ROM. In the new design, with the column select transistors at the top of the ROM, three of the four columns will be disconnected from the bit line. This greatly reduces the capacitance of the bit line that must be driven by the the pull up or row select transistors. Something else to be noted is that since the new ROM is not as high as the ROM in the old design, the length of the bit line routing will be shorter which also

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

helps contribute towards reducing the bit line capacitance. The reduction of all these capacitances, in combination with the timing change, are responsible for the increase in the performance of the ROM. These improvements are summarized in Table 1.

Capacitive Load for	in the first design	in the improved design	% reduction
row select lines	272 ff	267 ff	2%
column select lines	550 ff	116 ff	79%
bit lines	787 ff	290 ff	37%

Table 1. Capacitive Loads for Old and New Binary Design

Looking at the structural diagram in Fig. 21a again, one sees that at the output of the ROM there are no transmission gates as there are in the first design. Those gates were no longer necessary since it was possible to achieve the same effect by conditionally blocking the precharge and column select signals to the ROM. From Fig. 22, one can see that if the precharge signal is kept high and all the column select signals, low, then the bit line will be isolated from the ROM. This will allow some other circuitry to drive the bit line high or low without interference from the ROM. This eliminated 5 transmission gates and the routing of the signals needed to control them. The cost of doing this was 3 nand gates and two inverters. Although the silicon savings were not that great for this design change, it did help to reduce the capacitance siting

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

on the bit lines.

The last two sections that need to be discussed are the row and column decoders themselves. Both decoders are based on a transmission gate matrix, see Fig. 23. In this circuit, B_4 controls which column of transmission gates is enabled. For the column that is disabled, ie. the input to the pMOS transistors is high, all the nMOS transistors between the output and ground will be on. This will pull these signals to ground. In the other column of transmission gates, the nMOS transistor, from output to ground, will be off which allows the input to the transmission gate to pass through unaffected. Since the inputs to the two transmission gates are the complements of each other, only one output from this transmission gate matrix will be high at a time. A comparison between this structure and the one used in the first design, on the basis of number of transistors required, is given in Table II. Also included in the table is the number of transistors used in a nor based decoder.

From the table one can see that the transmission gate matrix decoder uses slightly fewer transistors than a nor based decoder. Usually, however, a transmission gate design takes up more area than the same number of transistors would in a normal CMOS arrangement. Comparing actual areas of the two types of decoders supports this with the nor design requiring about 10% less area. With the nor decoder, though, one will not get quite the same speed so once again,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

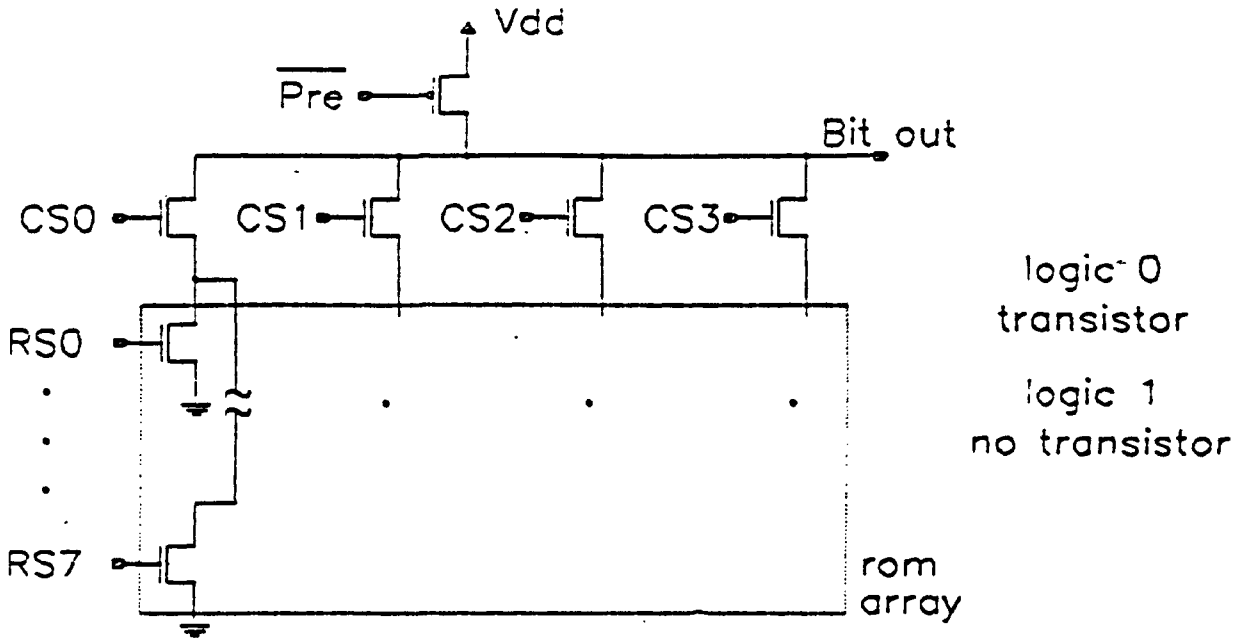


Figure 22. New Rom Structure

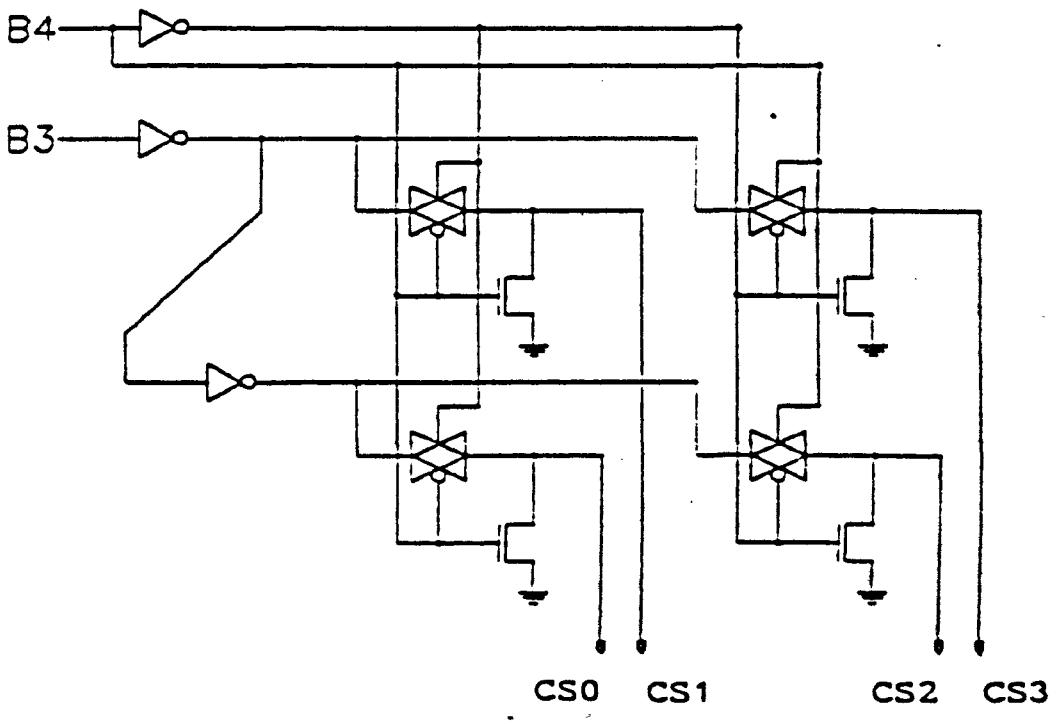


Figure 23. New Column Decoder Structure

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

		number of transistors
first design	2 * inverter	-> 2 * 2 = 4
	4 * nand + inverter	-> 4 * 6 = 24
		28
improved design	3 * inverter	-> 3 * 2 = 6
	4 * transmission gate + nMOS	-> 4 * 3 = 12
		18
minimum design	2 * inverter	-> 2 * 2 = 4
	4 * nor gate	-> 4 * 4 = 16
		20

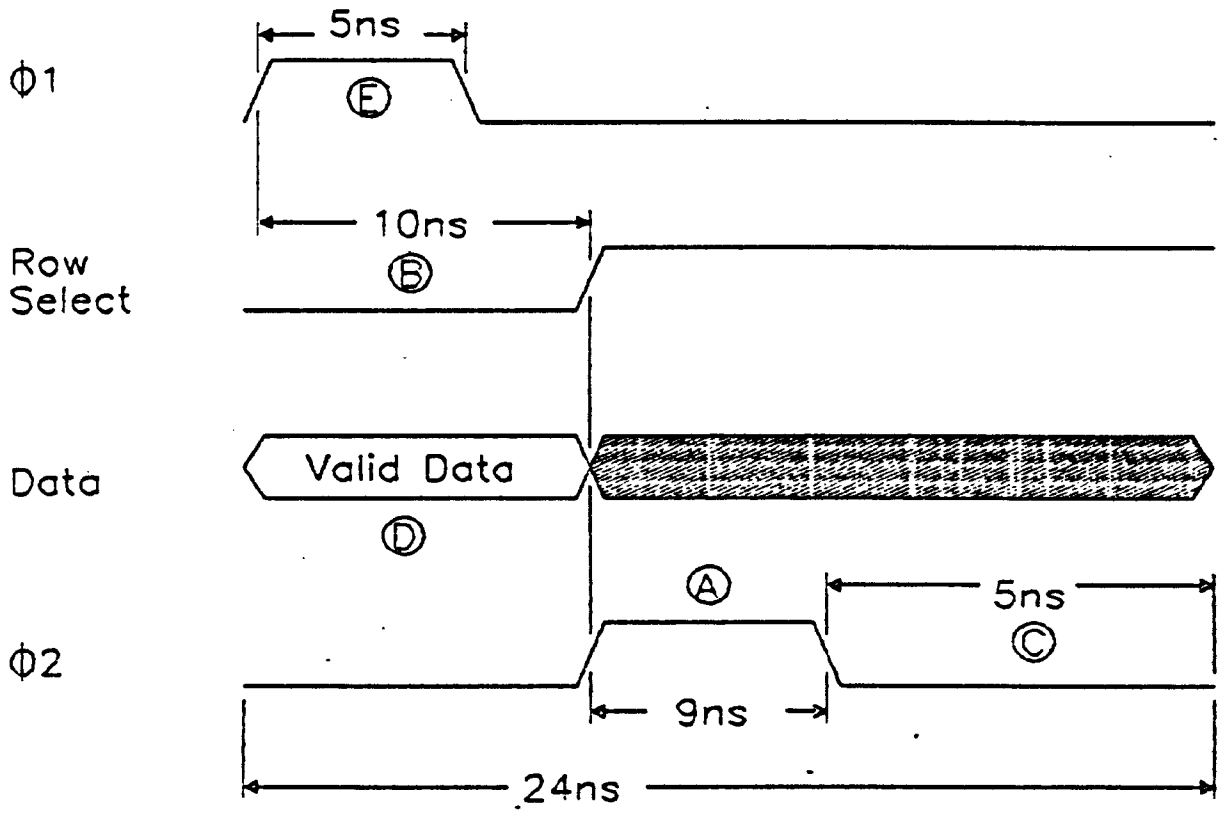
Table 2. Comparison of Decoder Sizes

there is a trade off between speed and area.

Another point that needs to be considered is that with the transmission gate decoder, only two large inverters, ie. those with increased drive capability, are required. In the other designs, four driver stages, one for each select line, would be needed. The same holds true for the row decoder, only two drivers are needed for the transmission gate design whereas eight would be needed for the other designs. The larger drivers or inverters take up more area than standard sized inverters, thus reducing the number called for helps to reduce the total area of the decoder.

Appendix III contains the layouts and simulation results for the new binary design. The timing diagram is shown in Fig. 24. Using the same approach to calculating power that was used for the first design, the estimated power dissipation for this design is approximately 3.2 mwatts when running at 20Mhz.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Ⓐ Precharge the bit lines of the rom

Ⓑ Evaluate Row Decoder

Ⓒ Evaluate the rom

Ⓓ Data valid from previous stage

Ⓔ Latch the data from the previous stage

Figure 24. Timing Diagram for Improved Binary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The ROMs in the first binary design and in the ternary design were programmed by hand. For the improved binary design, a rudimentary silicon compiler was written to program the ROM automatically. Due to the regular layout of the ROM, the program was fairly simple. The listing of the program can be found in Appendix V.

When the design was laid out, all the locations in the ROM were left empty, ie. no row select transistors were placed in the design. All that would be required to program the ROM would be to create a set of geometries representing the row select transistors and place it on top of the empty ROM. The program that was written creates just such a set of geometries.

To be specific the program first creates a basic cell for one row select transistor. Then, given the modulus for the ROM, the program generates the proper CIF codes, Caltech Intermediate Format, that place the transistors cells at the appropriate locations. Once this data file is generated it can be read into the design data base and placed on top of the empty ROM. This approach is obviously much faster and more accurate than the manual approach.

Another program that was put to use for the improved binary design is also included in Appendix V. One of the requirements that must be met when the designs are sent off to be fabricated is that all geometry must be centered about the origin. When one is actually doing the design work, it is easy to forget that the design must be centered. The process of centering the design after it has been completed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is tedious and error prone. The original motivation behind the program was simply to identify those sections of the design that were not centered.. It then developed into a utility program that would center the designs by itself. A number of other features were added to the program to make it more useful. The most important of these was the rounding off of the vertices of the polygons. Some of the design software being used did not always produce coordinates that were the requisite multiples of 50. When this occurred errors were detected in the design that did not actually exist. In order to avoid this problem, the program checks all vertices and rounds to the nearest multiple of 50. In addition to this, the program checks on how many levels there are in the design hierarchy, determines the size of all of the cells and provides statistics on the number of times a cell is called and the number of polygons the cell contains.

In writing this program it was necessary to understand the CIF code format and to be able to extract the appropriate information from the codes, manipulate the data and generate a new set of CIF codes. In doing this, the program frees the designer from worrying about centering designs and checks certain parameters to help determine whether or not the design is suitable for submission for fabrication.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

V. The 3 Trit Ternary RNS Adder

The equations that are pertinent to the operation of the ternary rns adder were discussed in section II. From these discussions it was demonstrated that whereas the binary design required five modules to be cascaded, in the ternary version, only three modules are needed. Before continuing, it is important to note that the ternary design will only allow the numbers 0 through 26 to be represented in 3 trits. In contrast, the binary design can represent the numbers 0 through 31 using 5 bits. When comparing the two designs, this fact must be kept in mind, the two are not equivalent.

The basic structure of the ternary module is similar in function to that of the binary module. The two inputs, A and B, are fed into a set of latches which is used to construct a pipelined design. The B inputs also form the address into a ROM which can selectively be bypassed. The diagram in Fig. 25 represents the structure of the ternary module and, although considerably different from those of the previous two designs, does in fact share a similar architecture. The simulation results, layouts and circuit diagrams have been provided in Appendix IV.

One deviation from the approach of the two binary designs lies in the use of a ROM to replace the switching operations around the other ROMs. In the binary designs, when the A_i has a value of zero, and consequently no

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

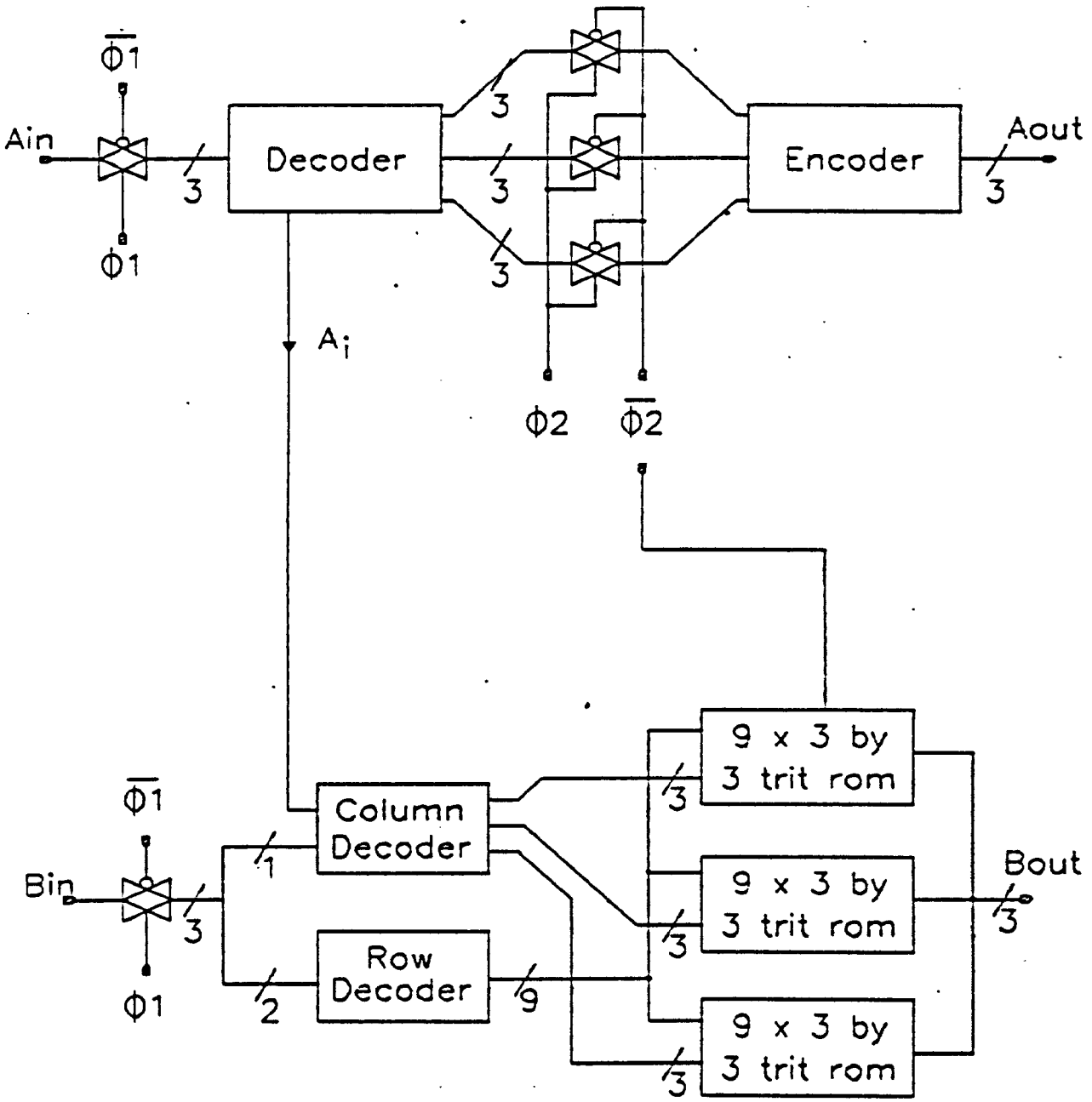


Figure 25. Structure of Ternary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

addition needs to be done, the input to the ROM is routed around the ROM to the output. With the ternary design, a third ROM was used for the case when A_i equals zero. This meant that the B inputs were decoded and fed to all three ROMs. The appropriate output was selected by activating the correct ROM. As it turns out, this approach is inferior to that used in the binary designs in that there is no increase in speed but rather an increase in the area required.

In the binary designs, the latches were based on either the clocked inverter or on a transmission gate followed by an inverter. In the ternary design, the principle behind the operation of the latches is the same as that behind the transmission gate latch. The only difference in this design is that instead of an inverter, either a decoder or an encoder is used.

The row and column decoding circuitry for this design both implement a 3 to 9 decoder. The transmission gate matrix method, as described in the section on the new binary adder, was used for both. The ternary to binary decoders generated the inputs to these circuits. The 9 row select signals were fed to each of the three ROMs. The 9 column select signals were broken into three groups of three, each group being routed to a separate ROM. Each of the three ROMs is configured as nine rows by three columns by three trits. The ROM structure is shown in Fig. 26. Just as in the new binary design, this design has the advantage that the columns select transistors isolate the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

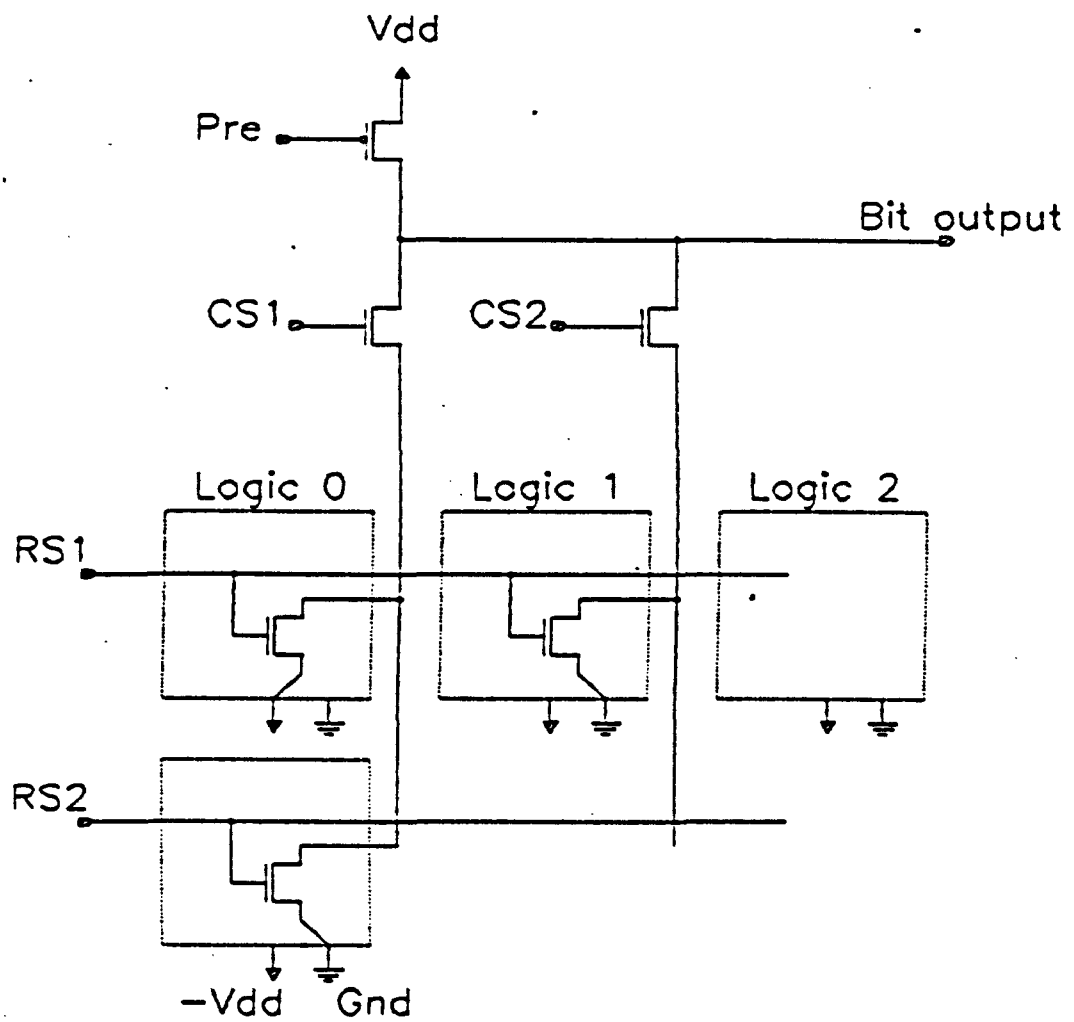


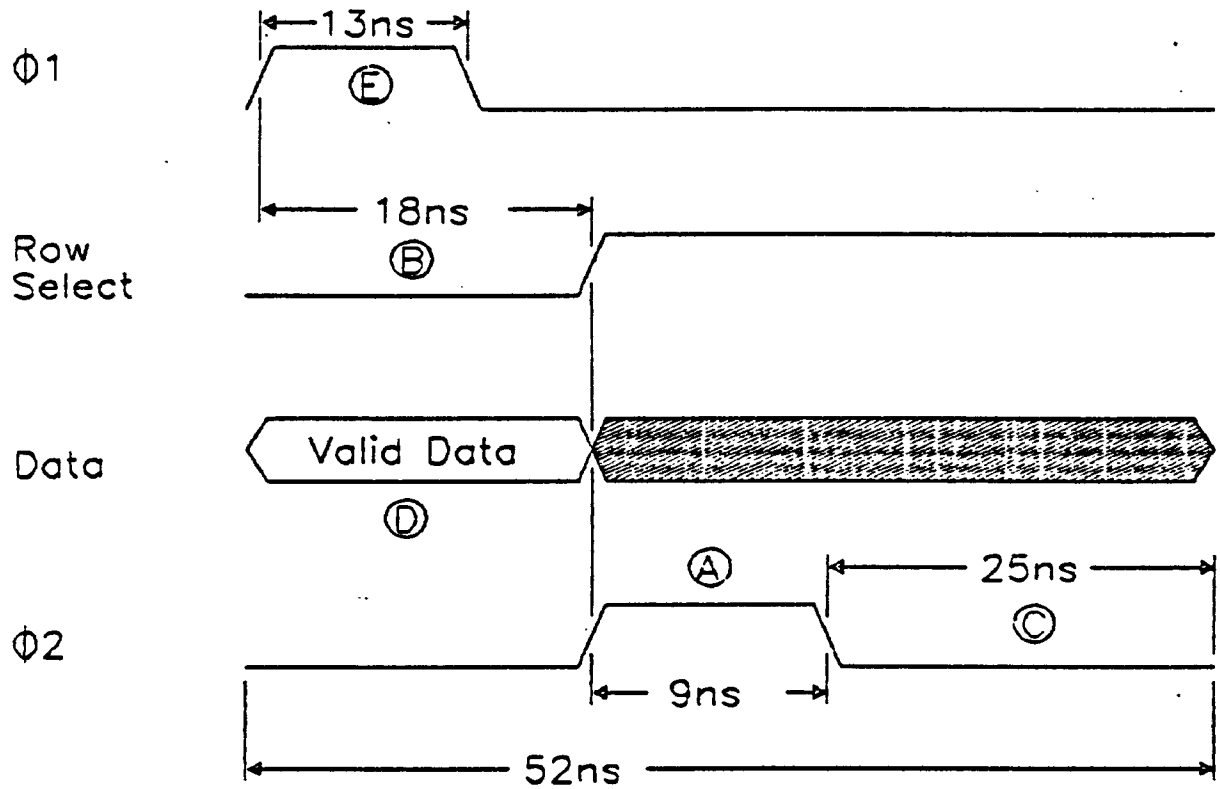
Figure 26. Internal Rom Structure for Ternary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

columns that have not been selected from the one that has. Since the column decoder only produces one high signal out of the nine, only one column will be activated. This reduces the capacitance of the bit lines significantly. In an effort to do the same with the row select lines, a slightly different tact was taken. Instead of feeding the row select signals directly to each ROM, they were sent to three sets of nine transmission gates, one set per ROM. The ROM that was to be accessed would have its corresponding set of nine transmission gates turned on while the other two ROMs would have their sets of transmission gates turned off. This, once again, isolated the gate and routing capacitance of the row select lines of the two ROMs that were not in use from that of the one ROM that was in use. This produced faster rise and fall times on the row select signals. It should be pointed out that in a future design, one that only had two ROMs, the reduction in the row select signal capacitance attributable to this technique would be marginal. For this reason, the row select signals would be fed directly to both ROMs and the transmission gates removed.

Data from the simulations in Appendix IV was used to generate the timing diagram shown in Fig. 27. Using the same technique for calculating dynamic power dissipation for an operating frequency of 20mhz and taking into account the static dissipation of the ternary inverters, it was determined that this circuit used approximately 5.9 mwatts per module.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Ⓐ Precharge the bit lines of the rom

Ⓑ Evaluate Row Decoder

Ⓒ Evaluate the rom

Ⓓ Data valid from previous stage

Ⓔ Latch the data from the previous stage

Figure 27. Timing Diagram for Ternary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

VI. Conclusions

As was stated in the Introduction, the main objective of this thesis was to determine whether there are any advantages to be gained by applying multi-valued logic design techniques to the high speed DSP circuits being developed at this University. If one were to evaluate the usefulness of a given design, the criteria would be, in decreasing order of significance;

a) operating speed
b) circuit area
c) power requirements.

The results of the design work discussed in the preceding sections are summarized in Table 3. Also included in the last column are the estimates for an improved ternary design. The figures for the new design were derived from the first ternary design taking into account the changes that would have been made had an improved design been implemented.

Foremost among these changes is the removal of the ROM for the A_i=0 case. This would lead to a reduction in the area required without a significant impact on the clock speed. Knowing the area of the ROM being removed and the area of the switching circuitry being added, one can estimate the size of the new design to be around 600 microns by 600 microns. In order to compare with the improved binary design, we should also take into account the fact

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Parameter	First Binary Design	Improved Binary Design	First Ternary Design	Improved Ternary Design
# of Modules	5	5	3	3
Area/Module (sq. mm)	499.2u x 510.6u = 0.255	421.2u x 377.4u = 0.159	601.2u x 748.2u = 0.450	600u x 540u = 0.324
Total Area (sq. mm)	1.274	0.795	1.349	0.972
Power/Module (mwatts)	2.2 @ 10Mhz	3.2 @ 20Mhz	5.9 @ 20Mhz	5.5 @ 20Mhz
Total Power (mwatts)	11.0	16.0	17.2	16.5
Clock Period (ns)	90	24	52	40
Frequency (Mhz)	11.1	41.7	19.2	25

Table 3. Comparison of Design Results

that the improved binary design was fabricated using a double metal process. This generally gives around a 10% reduction in area, [Irwi86], when compared with the same design done in single metal. To allow for this, the area of the new ternary design is modified to around 600u by 540u.

The overall power requirements of the new design would probably not change significantly from those of the old ternary design. The use of two ROMs without transmission gates on the row select lines would mean that the row select signal that is being driven, is fed to two ROMs thus twice as much capacitance would be charged and discharged. This would lead to an increase in the power being consumed. On the other hand, to counter balance this effect, the more dense double metal design will have smaller capacitances on the routing signals from one part of the chip to another, due to the shorter distances. These decreased capacitances would tend to cause a reduction in the power consumption.

One could also expect a small increase in the speed of the circuit as a result of the more dense circuitry and reduced capacitances. The estimates in the last column of Table 3 were based on these qualitative assessments.

If one were to use a process that provided depletion mode transistors or designer control over threshold voltages, then it would be possible to improve the ternary design further either by reducing the area, increasing the speed, reducing the power consumption or any combination of the above. It is quite unlikely, though, that one would be able to match the speed performance of the binary design.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

It is also worth noting that from the economic point of view, even though it might be feasible to make the ternary version competitive with the right kind of process, the fact that one cannot use a simple inexpensive process needs to be taken into account. This means that when weighing the pros and cons of a multi-valued approach, the relative costs of the processes required need to be added to the list of criteria used to evaluate the design.

To digress slightly for a moment, it is interesting to examine the problem of RNS multiplication. As mentioned in section III, it is possible to use the same basic module used in rns addition, for the operation of multiplication. If one uses the straight forward approach it turns out that for n signals one needs n^2 modules. Thus for the binary design one would require 25 modules and for the ternary design one would need only 9 modules. Comparing the areas, one sees that the binary multiplier would use 3.975 mm^2 of silicon whereas the ternary design would only use 2.916 mm^2. As it turns out, though, there is a technique for reducing the number of modules required for multiplication, the quarter square method [BaJu87], which benefits the binary design but not the ternary design. This technique breaks the multiplication into several other operations using:

$$a \times b = \frac{(a+b)^2 - (a-b)^2}{4}$$

The number of modules needed for this are 3(n+1), where n is the number of signal lines. This method requires fewer modules than the direct method when n is larger than three.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

For the binary case one would only need 18 modules versus the 25 of the direct method. For the ternary design, this approach would require 12 modules rather than the nine from the direct approach. Now the binary design would use only 2.862 mm^2 of silicon. Although this is less than that of the ternary design, the two are very close.

In [Star81] which has already been discussed in section I, the author makes a comparison between a quaternary 30 Kbit microcode ROM and a similar ROM designed using standard techniques. From these results, see Table 4, it was demonstrated that the multi-valued design required 31% less area than did the binary design. The access time of the multi-valued ROM was reported to be less than 100ns; unfortunately, no figures were given for the binary ROM.

In [SoEs86] the authors carried out a comparison between three different design approaches to a FIR Digital Filter those being; standard binary, RNS ROM table-look-up, and multi-valued RNS. The results of their comparison are shown in Table 5. This data shows that the multi-valued design takes up less area, consumes more power and is slower than the corresponding RNS ROM based design. It should be noted that the authors, themselves, admit that the data for the number of ICs for the RNS ROM table-look-up approach is suspect. Despite the fact that this number could be off by a factor of 3 or more, the data agrees, for the most part, with that obtained from the designs conducted for this thesis. It was shown by way of the multiplication example

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ROM Type / Parameter	Standard	Other*	2 Bits/ Cell
Area/Bit (sq. microns)	210	169	112.5
Area/Bit Savings (%)	0	20	46
Area Overhead per Bit (sq. microns)	95.0	92.0	96.6
Normalized Area Overhead	0.45	0.54	0.86
Area Savings (%)	0	13	31

* Cell Type used on MOSTEK MK36000

Table 4. Results from Two Bits Per Cell Rom Design [Star81]

Item	Binary Filter	RNS ROM Table-Look_Up Filter	MVL RNS Filter
Number of ICs	80	247	27
Power Consumption	61 watts	118 watts	130 watts
Maximum Sampling Rate	11.8 Mhz	40 Mhz	32 Mhz
Minimum Delay Time	170ns	144 ns	248 ns

Table 5. Results of FIR Digital Filter Design [SoEs86]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and the quaternary ROM, that the multi-valued approach could under different circumstances require more or less area than a binary design. It was also shown that the power requirements of the multi-valued design would be greater and the speed less than that of the binary design. These conclusions are born out by the data from the FIR filter design.

It is interesting that the two techniques, current mode ecl and voltage mode CMOS have produced the same basic results. This tends to confirm the premise that multi-valued circuits by their very nature will be slower than their binary counter parts, regardless of the technology being used.

To summarize the findings, then, assuming the criteria of speed, area and power, in that order, it would not be advantageous to implement high speed DSP circuits using multi-valued logic. In certain designs, particularly those that depend heavily on memory structures such as ROM or RAM, significant area savings can be made by using a multi-valued design. Once again, though, if the memory structure is in the critical path of the design, and high speed operation is important, then multi-valued designs should not be used.

One consolation, even though the results of the thesis are of a negative nature, is that the design work for the binary versions of the RNS addition module have proven useful to others working in the field of RNS arithmetic.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

References

[AgPu84] V.Agarwal, J.Pugsley, "Multiple-Valued Rom Output Circuits", Proceedings International Symposium of Multiple Valued Logic - 1984, pp. 224-231.

[BaAn84] P.Balla, A.Antoniou, "Low Power Dissipation MOS Ternary Logic Family", IEEE Journal of Solid-State Circuits, Vol. sc-19, No.5, pp.739-749.

[BaJu87] M.Bayoumi, G.A.Jullien, W.C.Miller, "A VLSI Implementaion of Residue Adders", IEEE Transactions on Circuits and Systems, Vol. CAS-34, No. 3, March 1987, pp.284-288.

[CuFr85] K.Current, D.Freitas, F.Edwards, "CMOS Quaternary Threshold Logic Full Adder Circuits", Proceedings ISMVL-85, pp.318-322.

[CuWh80] K.Current, L.Wheaton, T.Luich, D.Mow, "Characteristics of Integrated Quaternary Threshold Logic Full Adders", Proceedings ISMVL-80, pp.24-30.

[EtNa85] D.Etiemble, B.Nathegi, "A 4-Valued 1Kbyte Rom Designed with Lambda Rules", Proceedings ISMVL-85, pp.92-100.

[FrCu83] D.Freitas, K.Current, "A Quaternary Logic Encoder-Decoder Circuit Design using CMOS", Proceedings ISMVL-83, pp.190-195.

[Garn59] H.Garner, "The Residue Number System", IRE Transactions, June 1959, pp.140-147.

[HeMo84] A.Heung, H.Mouftah, "DECMOS - A Low Power Family of Three-Valued Logic Circuits for VLSI Implementation", Proceedings ISMVL-84, pp.120-124.

[HuSa83] J.L.Huertas and G.Sanchez-Gomez, "Synthesis of 3-Valued Unary Operators in CMOS Integrated Technology", Int.J.Electronics, Vol.56, No.5, 1984, pp.714-726.

[HuSm83] M.Hu, K.Smith, H.Mouftah, "Low Power 2-of-3 Valued CMOS Self-checking Circuits", Proceedings ISMVL-83, pp.64-69.

[Irwi86] George Irwin, BNR, Ottawa, personal communication, December, 1986.

[KaHi84] M.Kameyama, T.Higuchi, "A New Architecture of a Pipelined Image Processor Based on Quaternary Logic Circuits", Proceedings ISMVL-84, pp. 92-97.

[LaEt84] E.Lavelle, D.Etiemble, "Improved Sense Amplifiers for 4-Valued Roms", Proceedings ISMVL-84, pp.232-239.

[McCl80] Edward McCluskey, "Logic Design of MOS Ternary Logic", Proceedings ISMVL-80, pp.1-5.

[McCl82] Edward McCluskey, "A Discussion of Multiple-Valued Logic Circuits", Proceedings ISMVL-82, pp.200-205.

[MoGa84] H.Mouftah, A.Garba, "VLSI Implementation of a 5 Trit Full Adder", IEE Proceedings, Vol 131, Pt.G, No.5, October 1984, pp.214-220.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[MoHe84] H.T.Mouftah, A.Heung, L.Wong, "QTC-1: A CMOS Ternary Computer", Proceedings ISMVL-84 Winnipeg, May 1984, pp.125-132.

[RiNa84] D.Rich, K.Naiff, K.Smalley, "A Four State Rom Using Multilevel Process Technology", Proceedings ISMVL-84, pp.174-179.

[ShAb82] S.Shaba, M.Abdul-Karim, "Quaternary Logic Techniques Applied to Digital Waveform Synthesis", Proceedings ISMVL-82, pp. 1-3.

[SoEs86] M.Soderstrand, R.Escott, "VLSI Implementation in Multiple-Valued Logic of a FIR Digital Filter Using Residue Number System Arithmetic", IEEE Transactions on Circuits and Systems, Vol.Cas-33, No.1, Januray 1986, pp.5-25.

[SzTa67] N.Szabo, R.Tanaka, "Residue Arithmetic and its Application to Computer Technology", McGraw Hill, New York, 1967.

[Tahe87] M.Taheri, personal communication, December 1986.

[TaJM87a] M.Taheri, G.A.Jullien, W.C.Miller, "High Speed Signal Processing Using Systolic Arrays", submitted for publication to IEEE Journal of Selected Areas in Communications.

[TaJM87b] M.Taheri, G.A.Jullien, W.C.Miller, "Systolic Rom Arrays for Implementing RNS FIR Filters", Proceedings International Conference on Acoustics, Speech and Signal Processing", 1987, pp.771-774.

[WhCu83] L.Wheaton, K.Current, "Quaternary Threshold Logic Full Adder Circuit with Complementary Inputs", Int. J. Electronics, Vol.56, No.4, 1984, pp.539-545.

[ZuAf85] C.Zukeran, C.Afuso, M.Kameyama,T.Higuchi, "Design of New Low-Power Quaternary CMOS Logic Circuits Based on Multiple Ion Implants", Proceedings ISMVL-85, pp. 84-90.

Bibliography

M.Annaratone, "Digital CMOS Circuit Design", Kluwer Academic Publishers, Boston, 1986.

P.Gray, R.Meyer, "Analysis and Design of Analog Integrated Ciruits", John Wiley & Sons, New York, 1984.

L.A.Glasser, D.W.Dobberpuhl, "Design and Analysis of VLSI Circuits", Addison Wesley, Don Mills, Ontario, 1985.

R.Troutman, "Latchup in CMOS Technology", Kluwer Academic Publishers, Boston, 1986.

N.Weste, K.Eshraghian, "Principles of CMOS VLSI Design, Addision Wesley, Don Mills, Ontario, 1985.

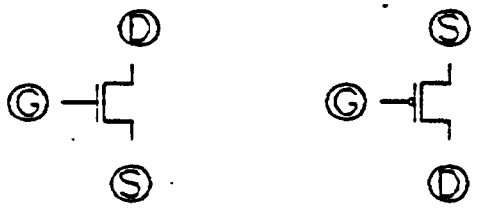
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix I

MOS Transistors: Symbols and Equations

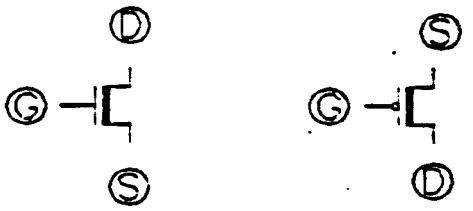
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Enhancement



N channel P channel

Depletion



N channel P channel

Transistor Terminals

Ⓓ – Drain

Ⓖ – Gate

Ⓢ – Source

Region of Operation	Drain to Source Current for nmos devices
Linear	$I_{DS} = B\{(Vgs-Vt)Vds-\frac{Vds^2}{2}\}$
Saturation	$I_{DS} = \frac{B}{2}(Vgs-Vt)^2$
Cutoff	$I_{DS} = 0$

Where: Vt = threshold voltage

Vgs = Gate to Source voltage

Vds = Drain to Source voltage

B = MOS transistor gain

$= Kp\frac{W}{L}$

MOS Transistors: Symbols and Equations

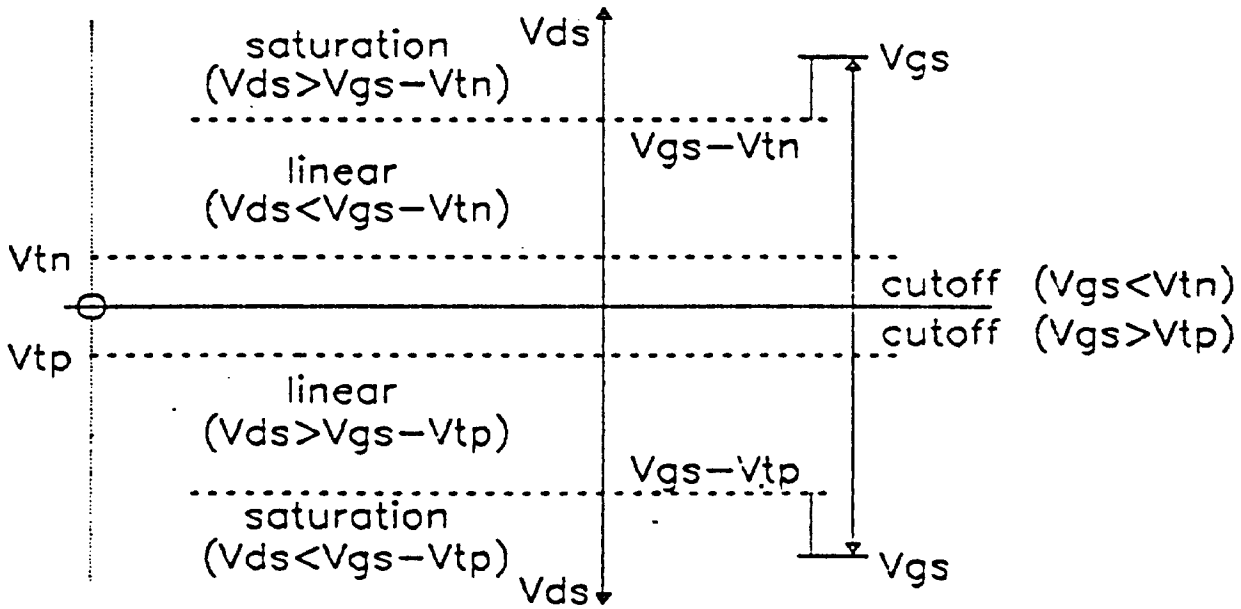
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Kp = process gain, a function of the fabrication process

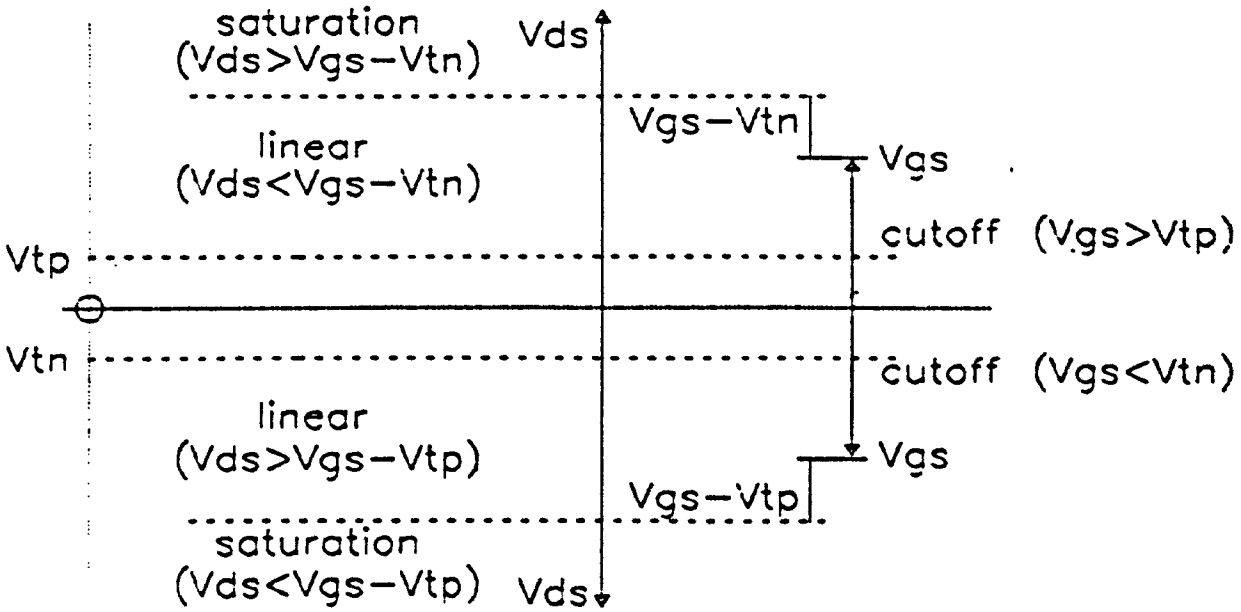
W = width of the gate region

L = length of the gate region

Enhancement Mode Transistors



Depletion Mode Transistors



Regions of Operation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix II

First Binary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix II. The First Binary Design

This Appendix contains the schematics, layouts and simulation results for the first binary design. In Fig. A.1 a legend of the layers used in the design is shown. The fill patterns and line styles indicated in this figure were used in the generation of the layouts included in Appendices II through IV.

Fig. A.2 lists the parameters that were used for the enhancement mode MOS transistors in the various Spice simulations that were conducted. All of the Spice decks that have been provided in the Appendices have had the MOS definition section removed since it was common to all.

The pin outs for the first binary design are shown in Fig. A.3. The plot of the complete chip that was actually submitted can be found after the schematics. A block diagram is also included to show the names of the blocks that were used in the design. A detailed plot of the main module is provided following this. A block diagram also accompanies this layout. The names on this diagram refer to the subsections of the design that are shown in the remainder of this Appendix.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

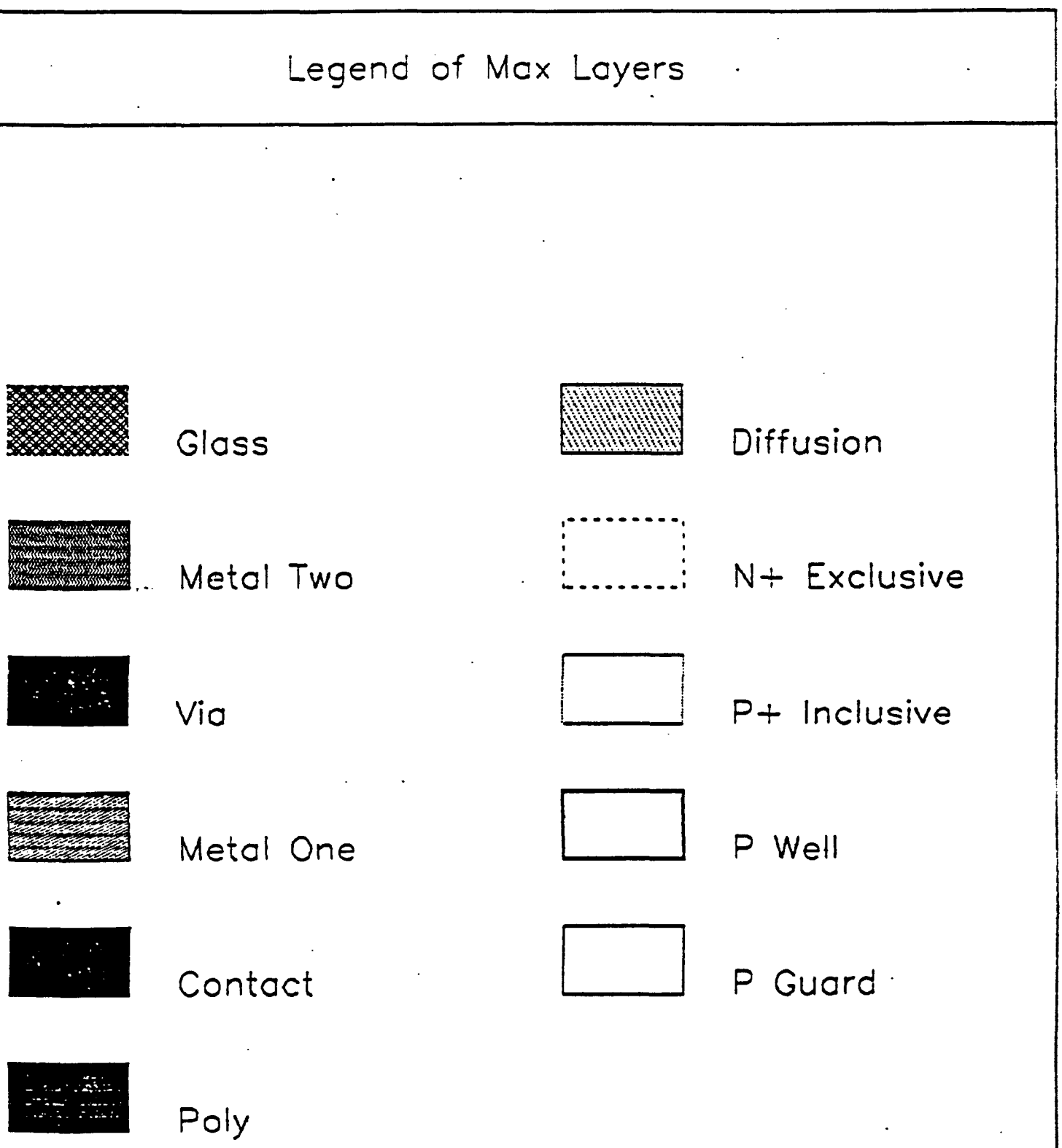


Figure A.1. Layer Line Styles and Fill Patterns

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
*----------------------
* MOS TRANSISTOR MODELS
*----------------------
* DRAIN GATE SOURCE SUBSTRATE MOD1 L=3U W=3U
* nmos enhancement
.OPTIONS DEFW=3U DEFL=3U DEFAD=81P DEFAS=81P NOMOD
.MODEL NMOSE NMOS LEVEL=3 KP=50.0E-6 VTO=0.7 TOX=5E-8 GAMMA=1.1
+PHI=0.6 LAMBDA=0.01 RD=40 RS=40 PB=0.7 CGSO=3E-10 CGBO=5E-10
+CGDO=3E-10 RSH=25 CJ=44E-5 MJ=0.5 CJSW=4E-10 MJSW=0.3 JS=1E-5
+NSUB=1.7E16 XJ=6E-7 LD=3.5E-7 UO=775 VMAX=1E5 THETA=0.11
+ETA=0.05 KAPPA=1
* pmos enhancement
.MODEL PMOSE PMOS LEVEL=3 KP=16.0E-6 VTO=-0.8 TOX=5E-8 GAMMA=0.6
+PHI=0.6 LAMBDA=0.03 RD=100 RS=100 PB=0.6 CGSO=2.5E-10 CGBO=5E-10
+CGDO=2.5E-10 RSH=80 CJ=15E-5 MJ=0.6 CJSW=4E-10 MJSW=0.6 JS=1E-5
+NSUB=5E15 XJ=5E-7 LD=2.5E-7 UO=250 VMAX=0.7E5 THETA=0.13
+ETA=0.3 KAPPA=1
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Pin #	I/O	Function
1	I	Gnd
2	O	B2
3	O	B1
4	O	B0
5	O	B3
6	O	B4
7	I	Precharge bar
8	I	Phi1 bar
9	I	Phi1
10	I	VDD
11	I	Phi2
12	I	Phi2 bar
13	I	A4
14	I	A3
15	I	A2
16	I	A1
17	I	A0
18	I	B2
19	I	B1
20	I	B0
21	I	B3
22	I	B4

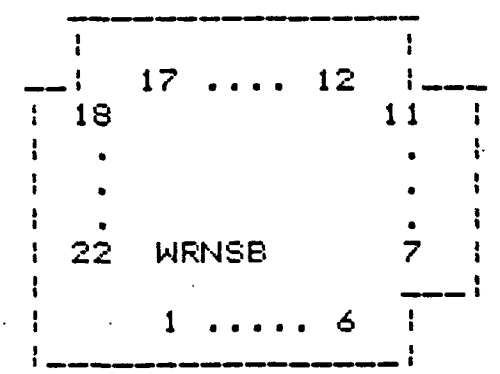
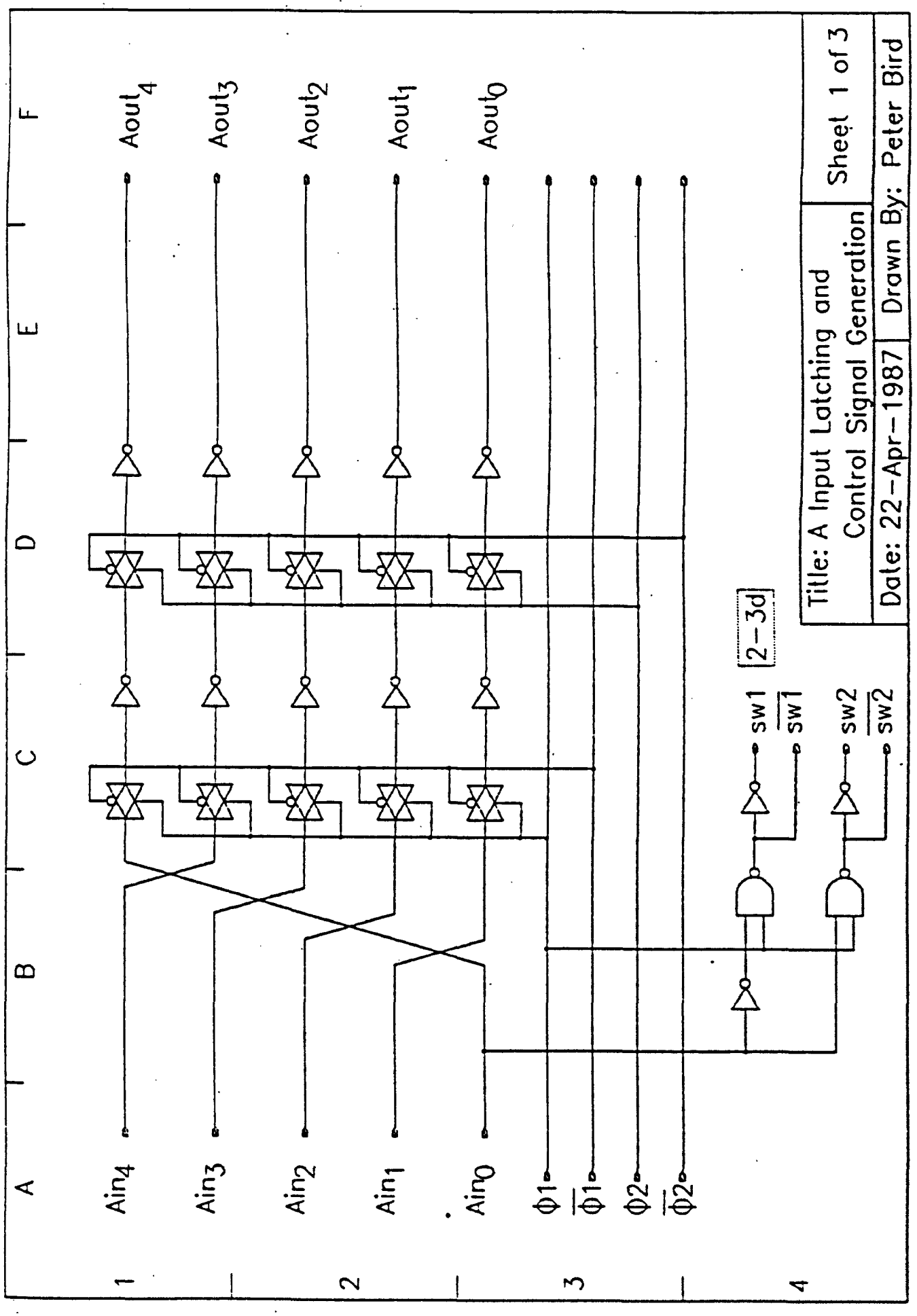
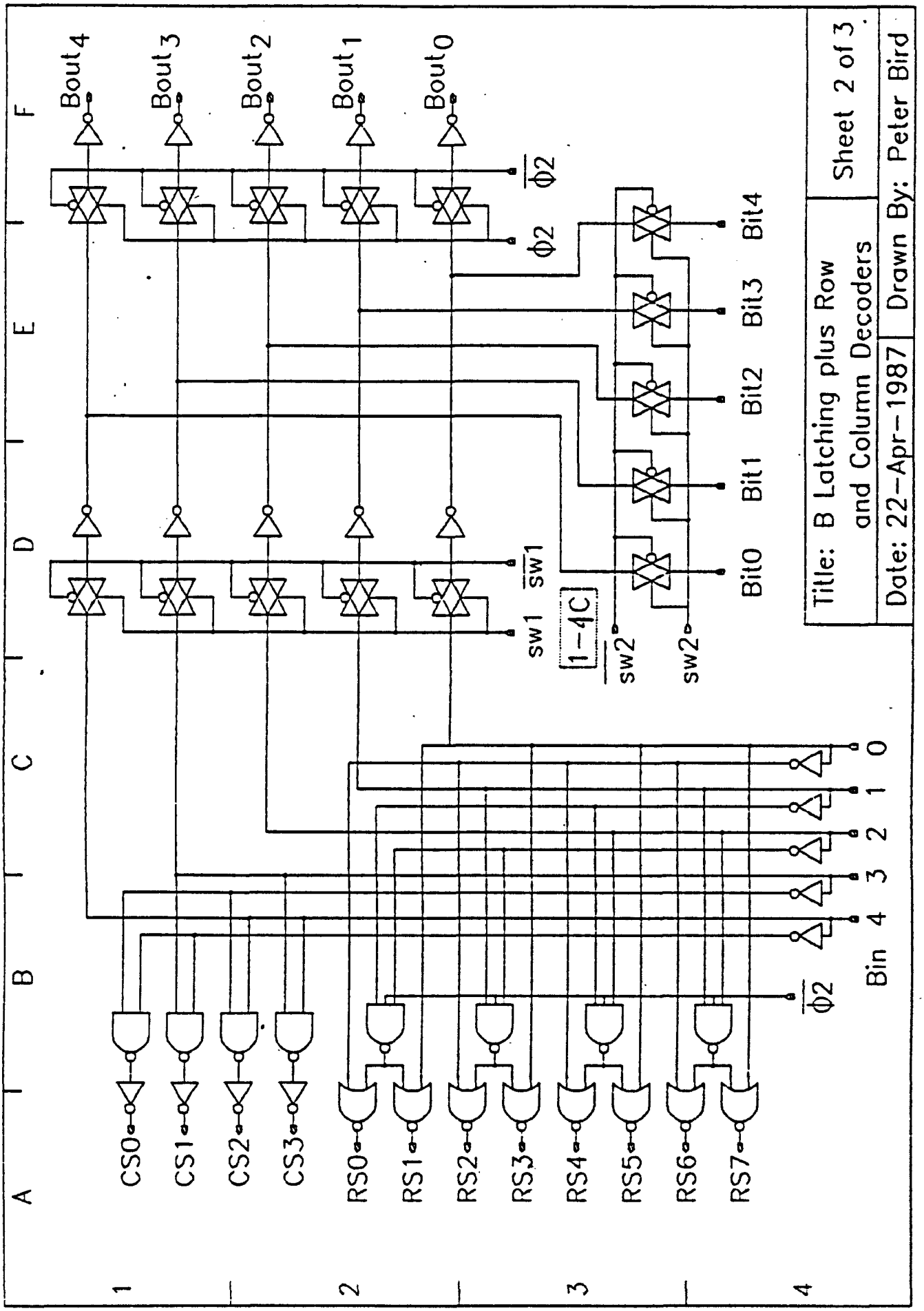


Figure A.3. Pinouts for the First Binary Design

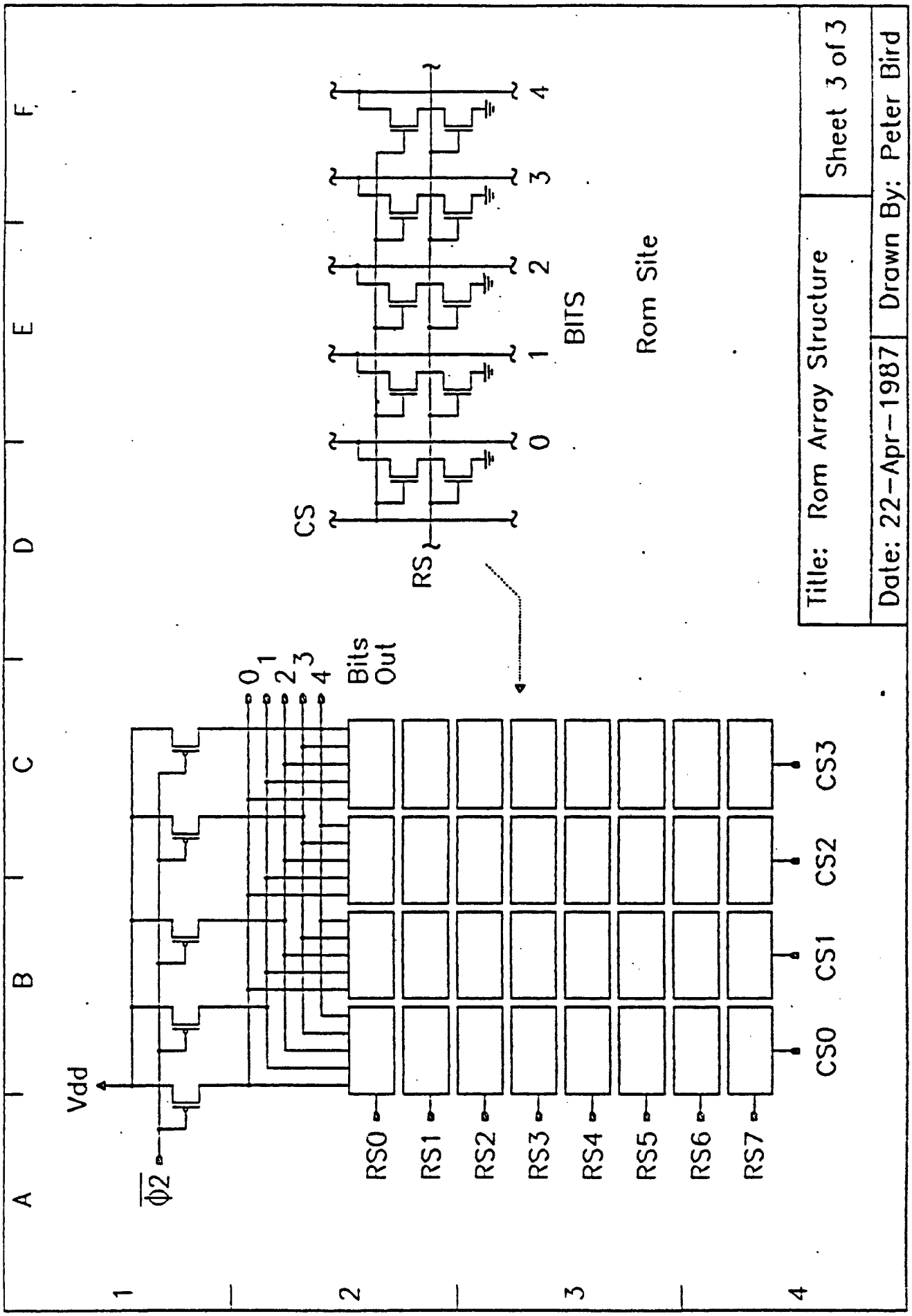
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



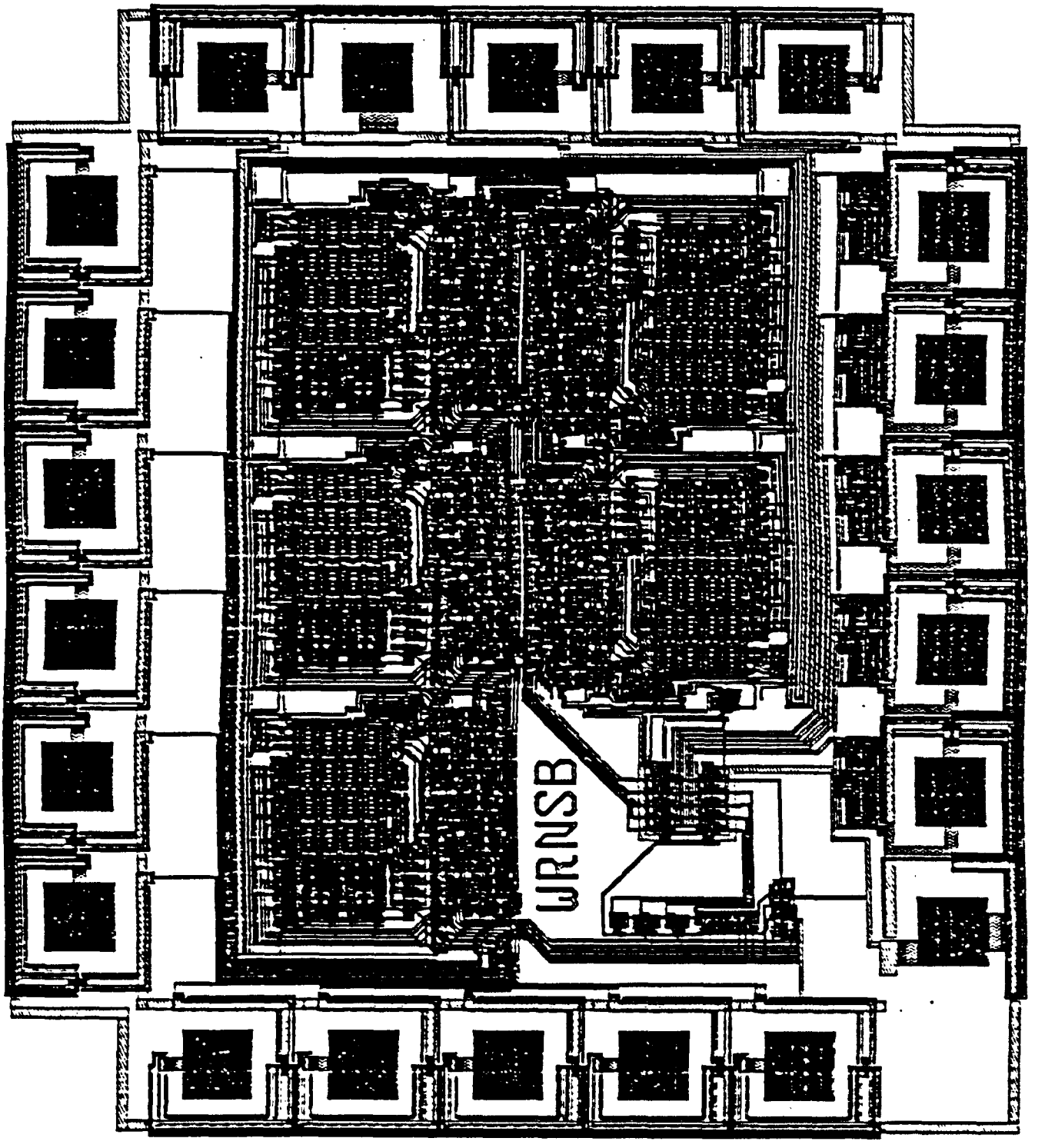
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



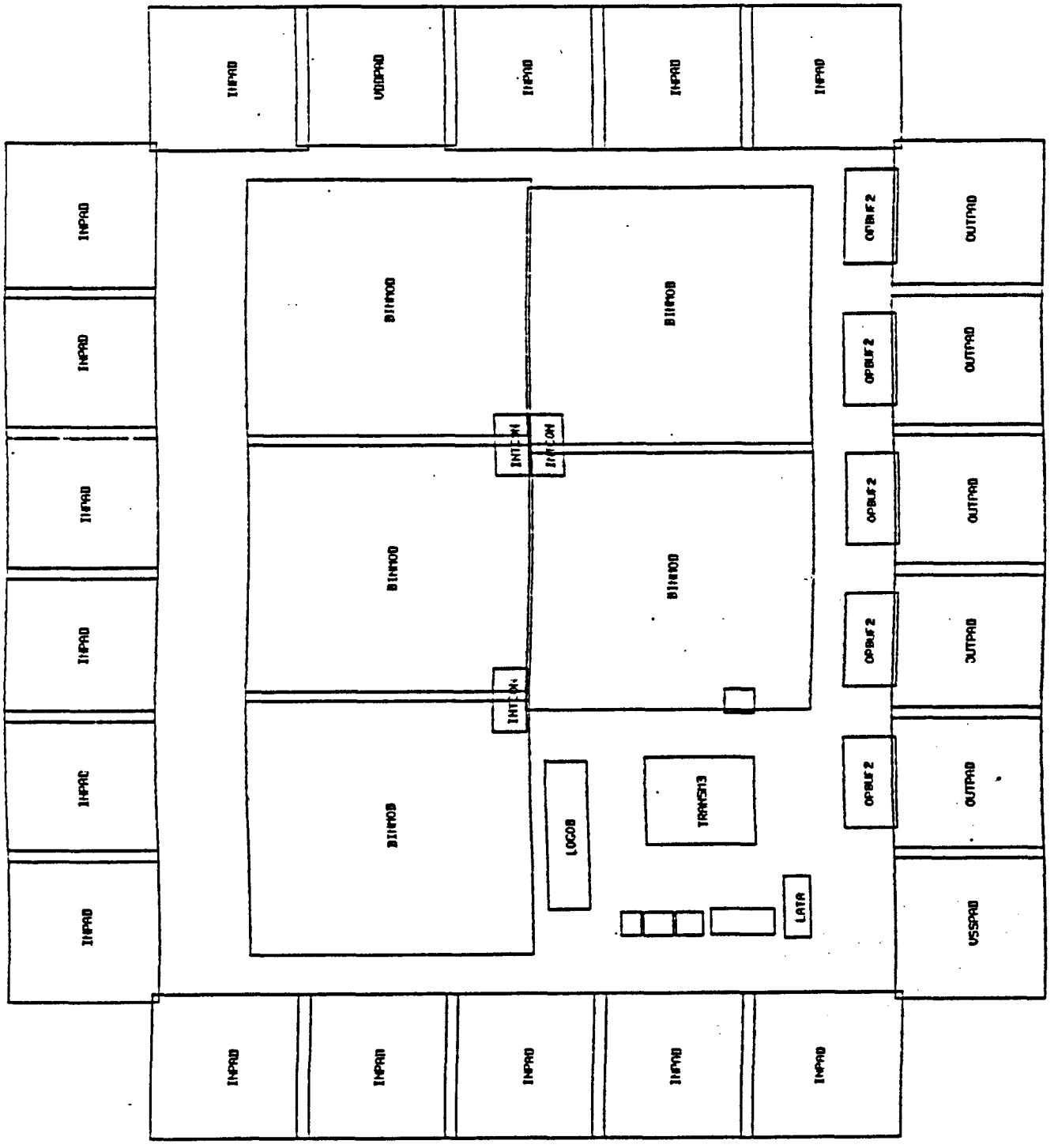
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



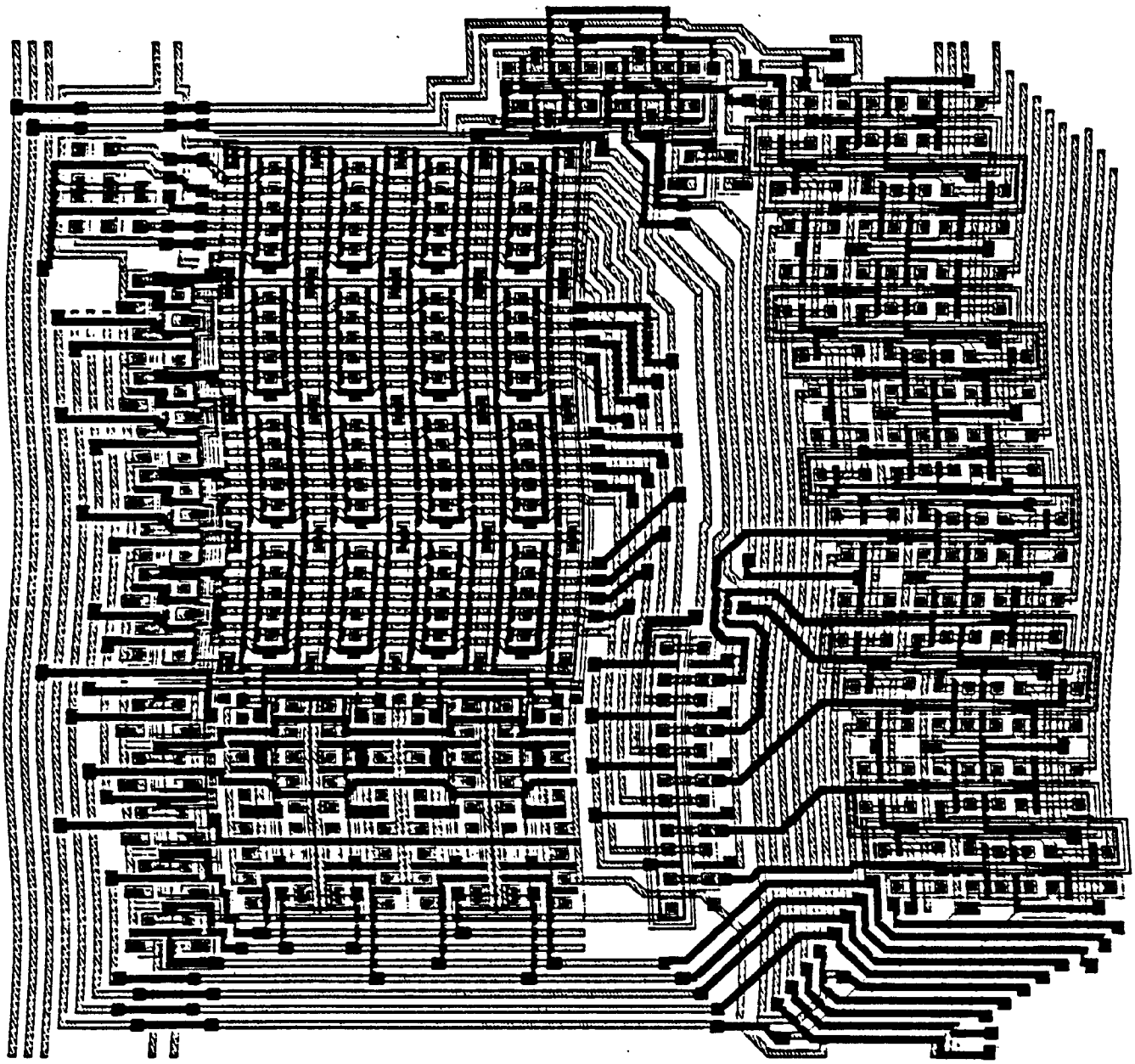
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



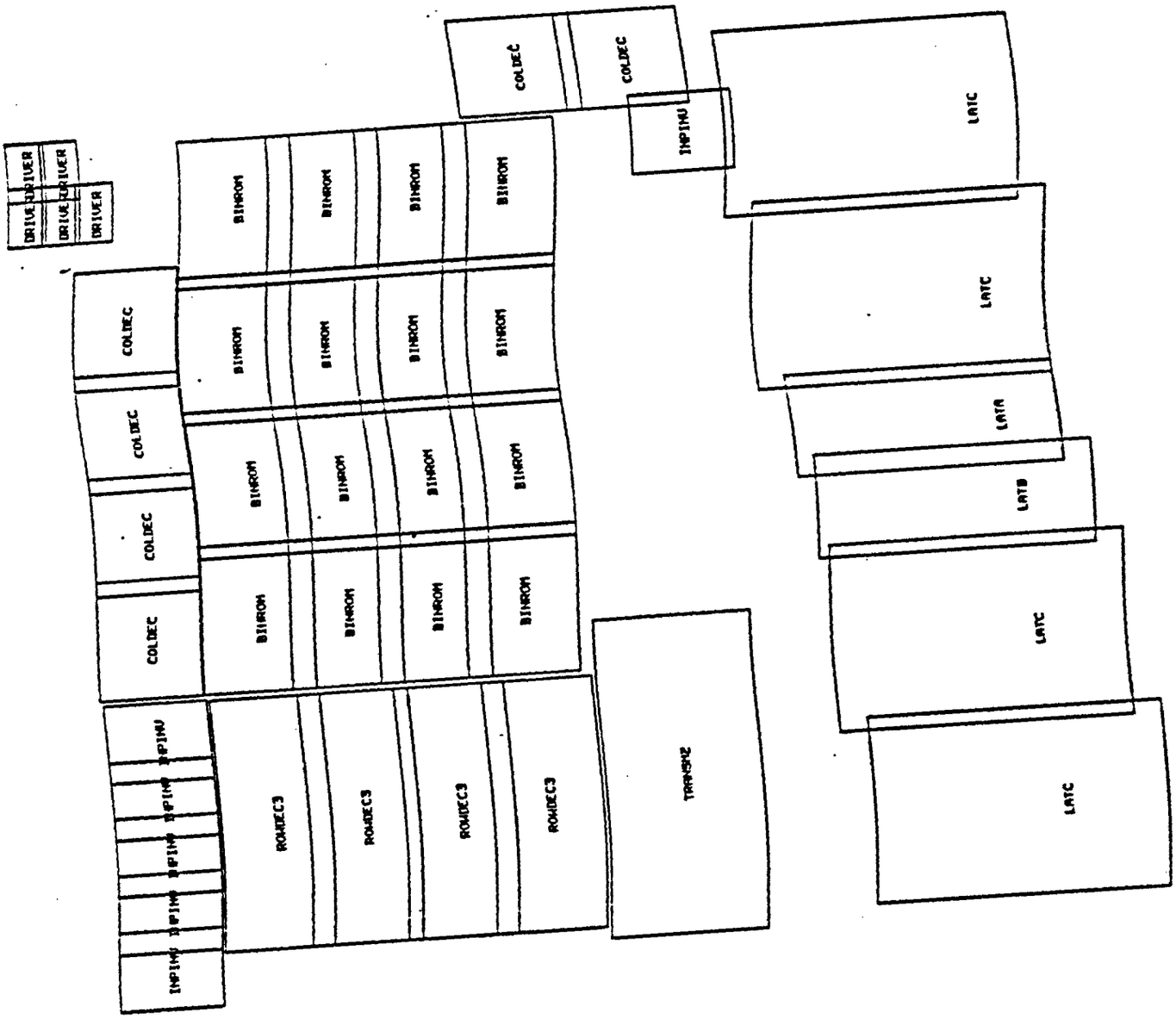
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



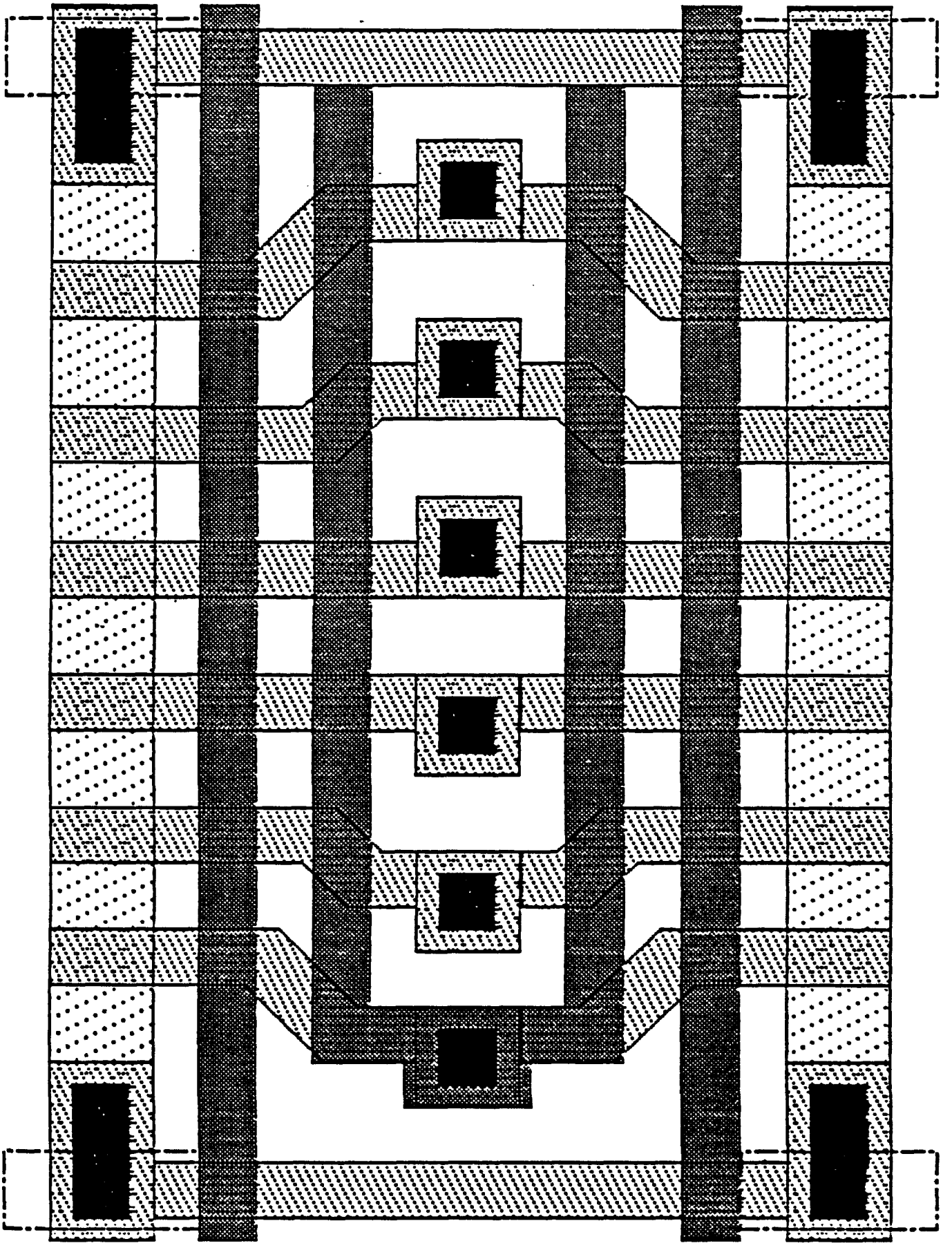
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



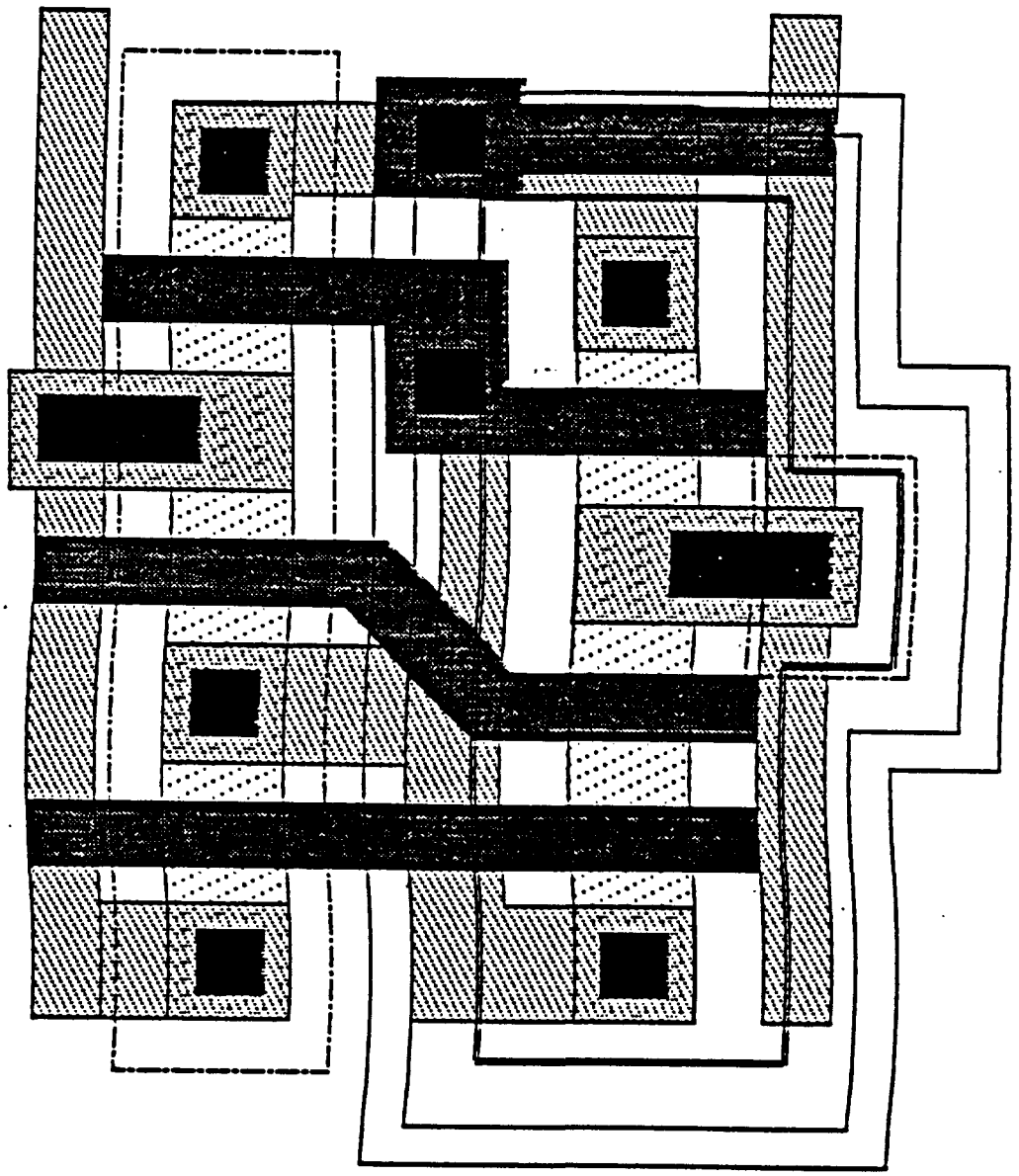
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

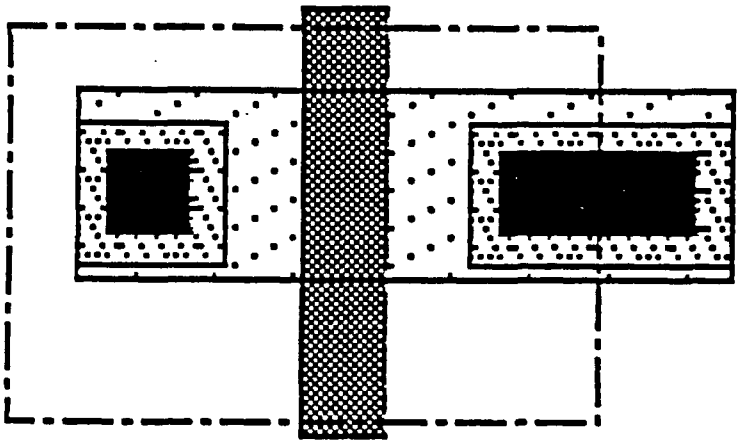


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



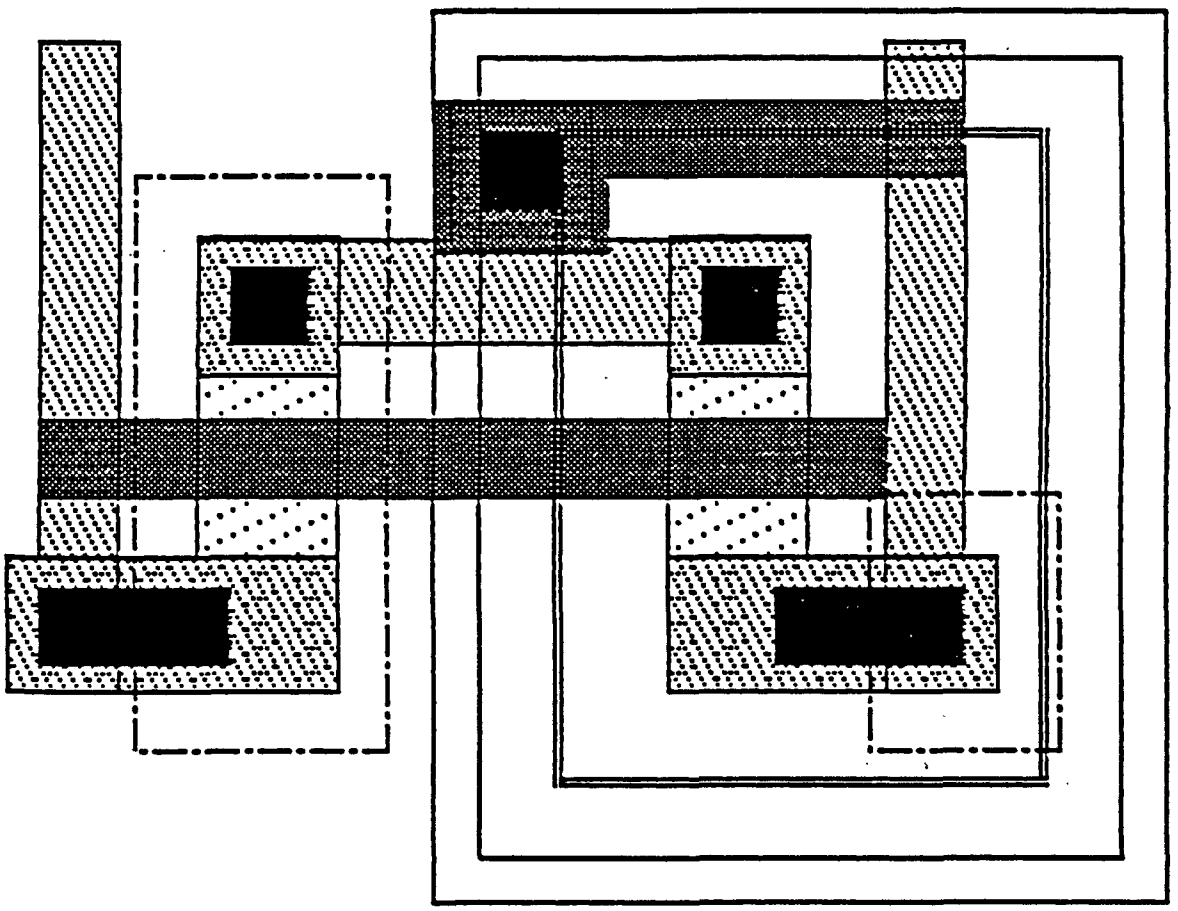
COLDEC

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



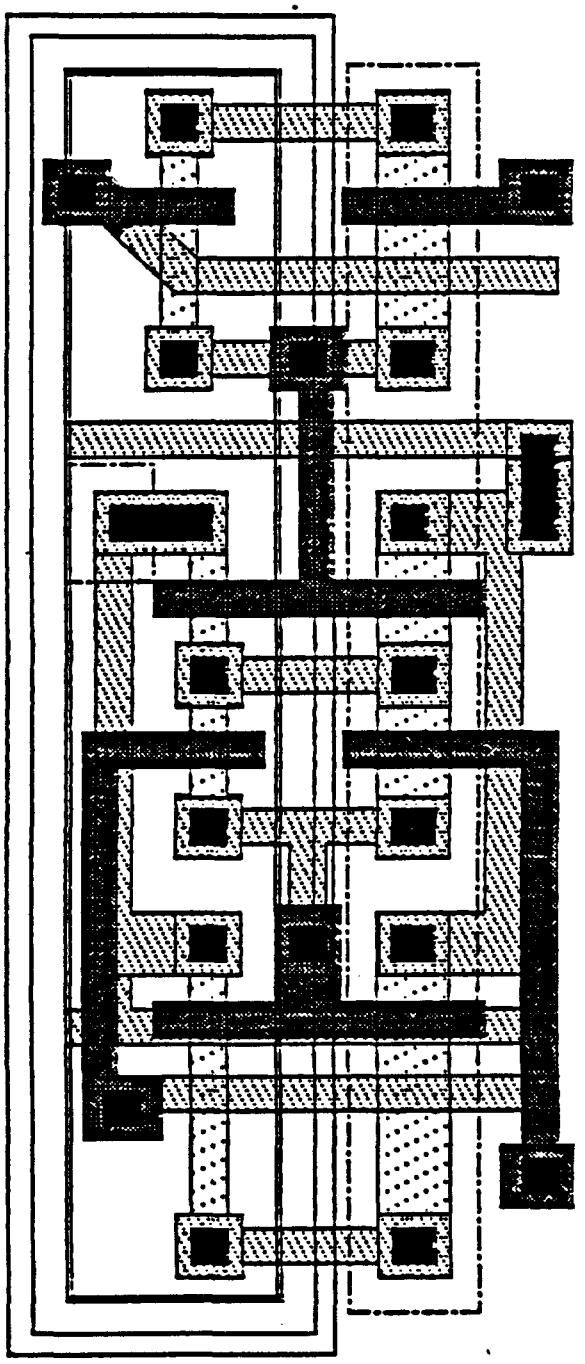
DRIVER

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



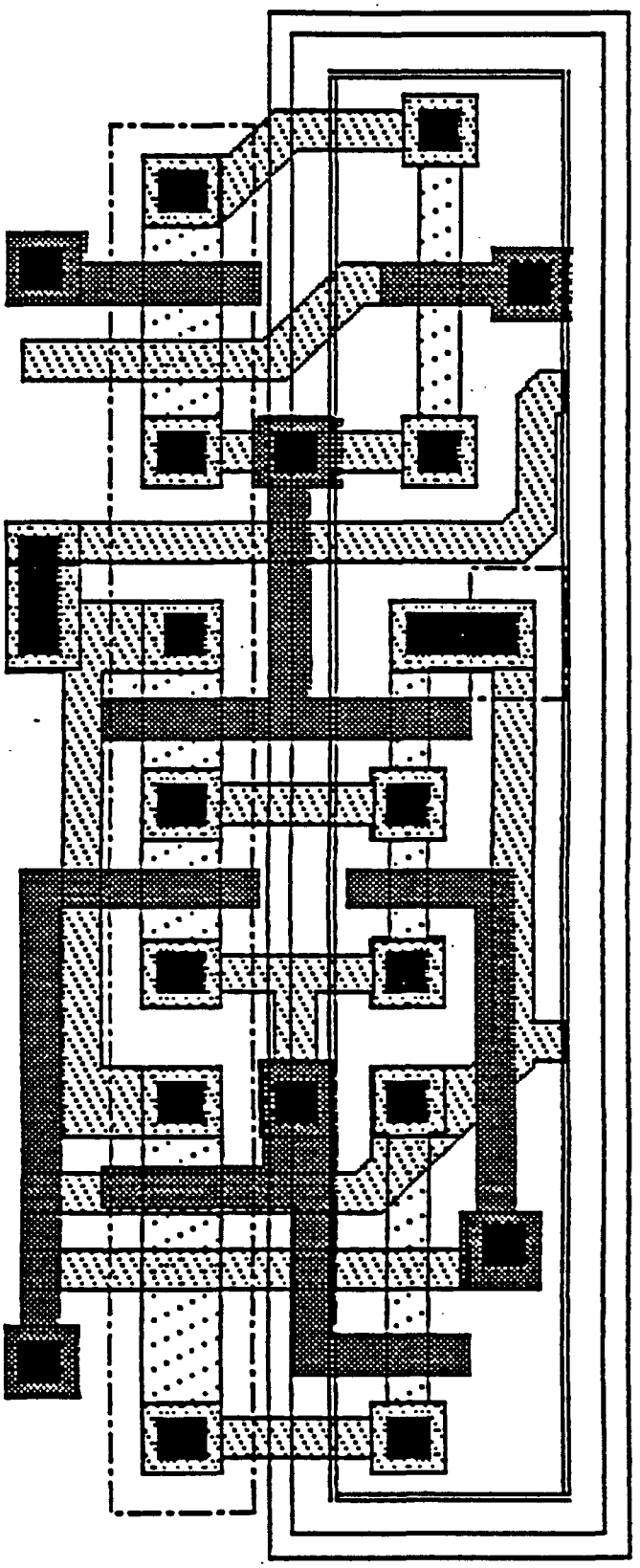
INPINV

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



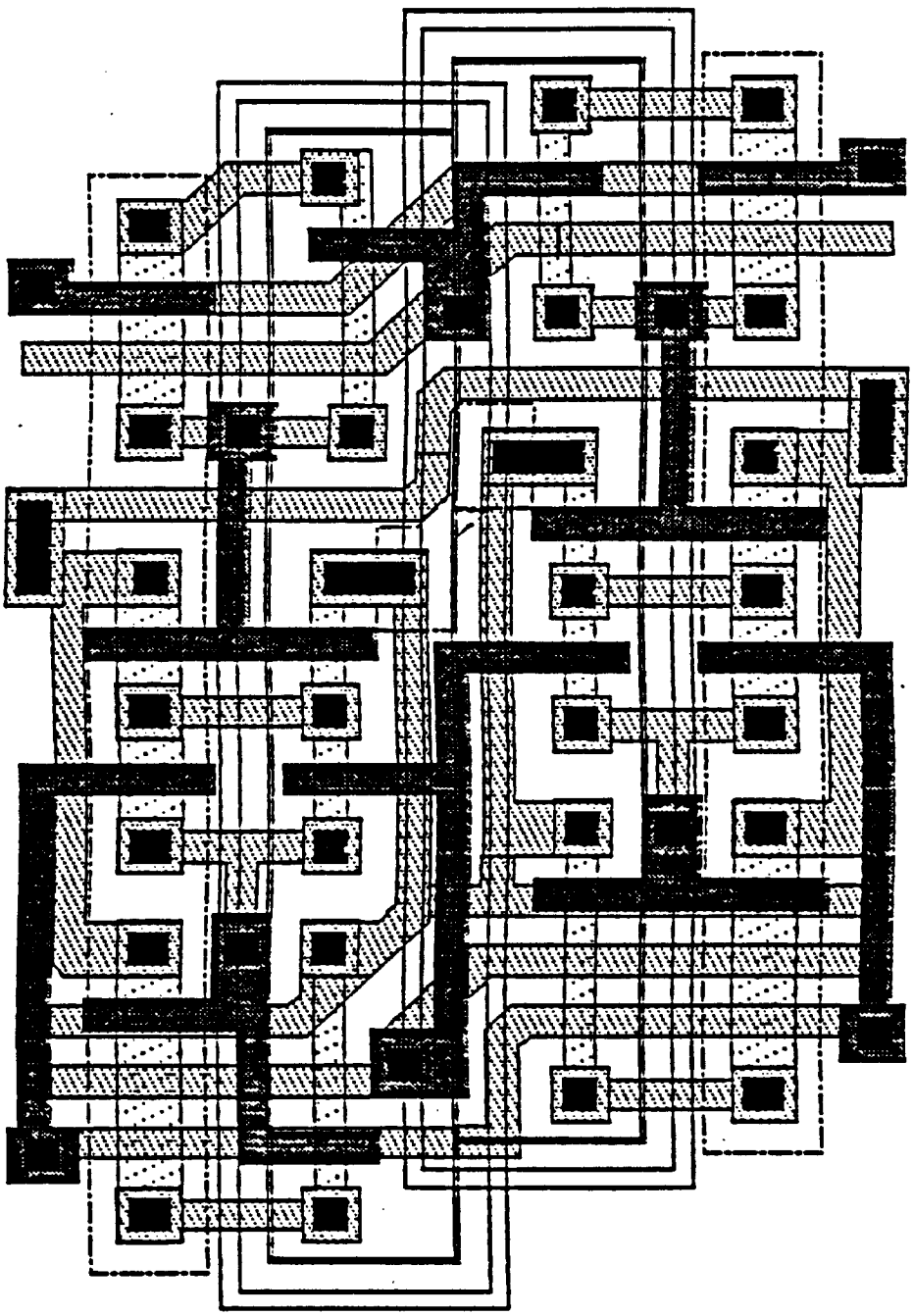
LATA

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



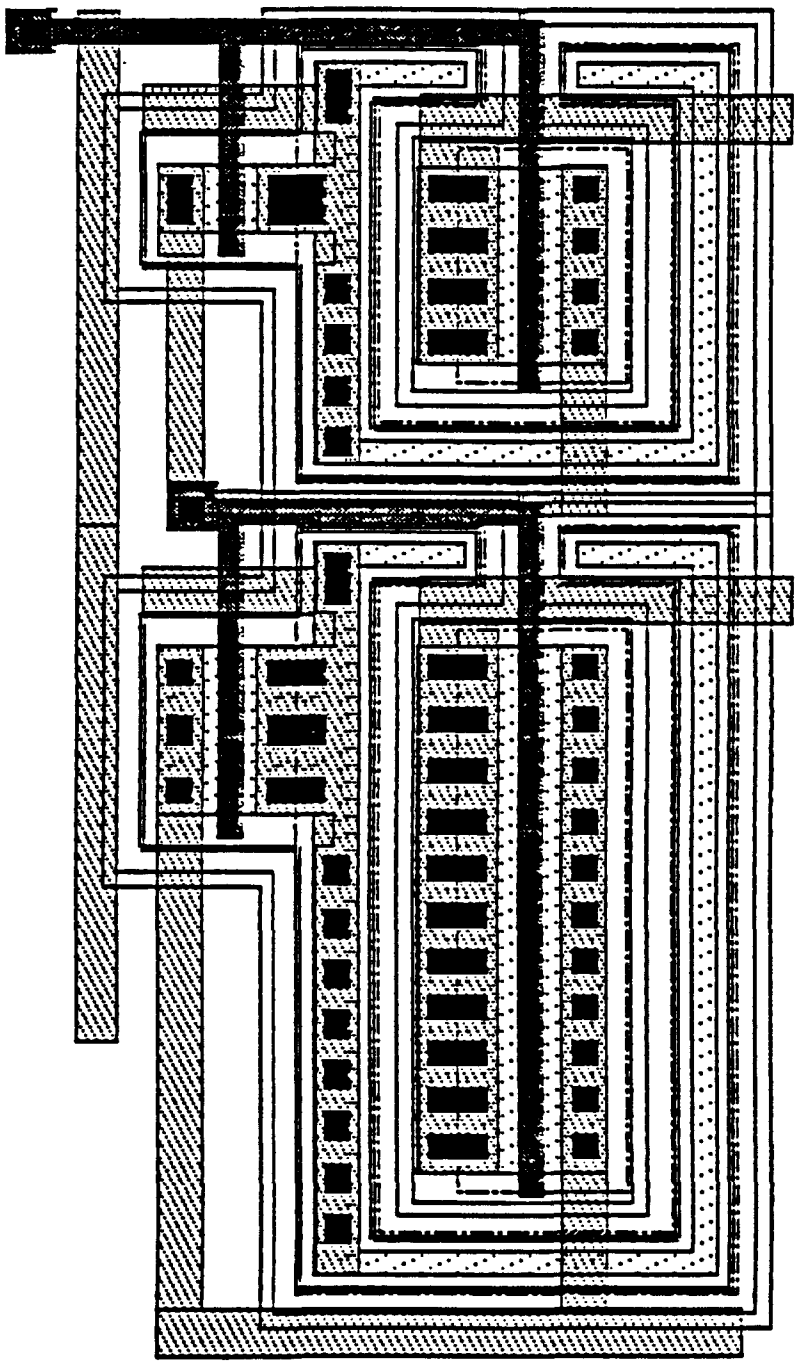
LATB

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



LATC

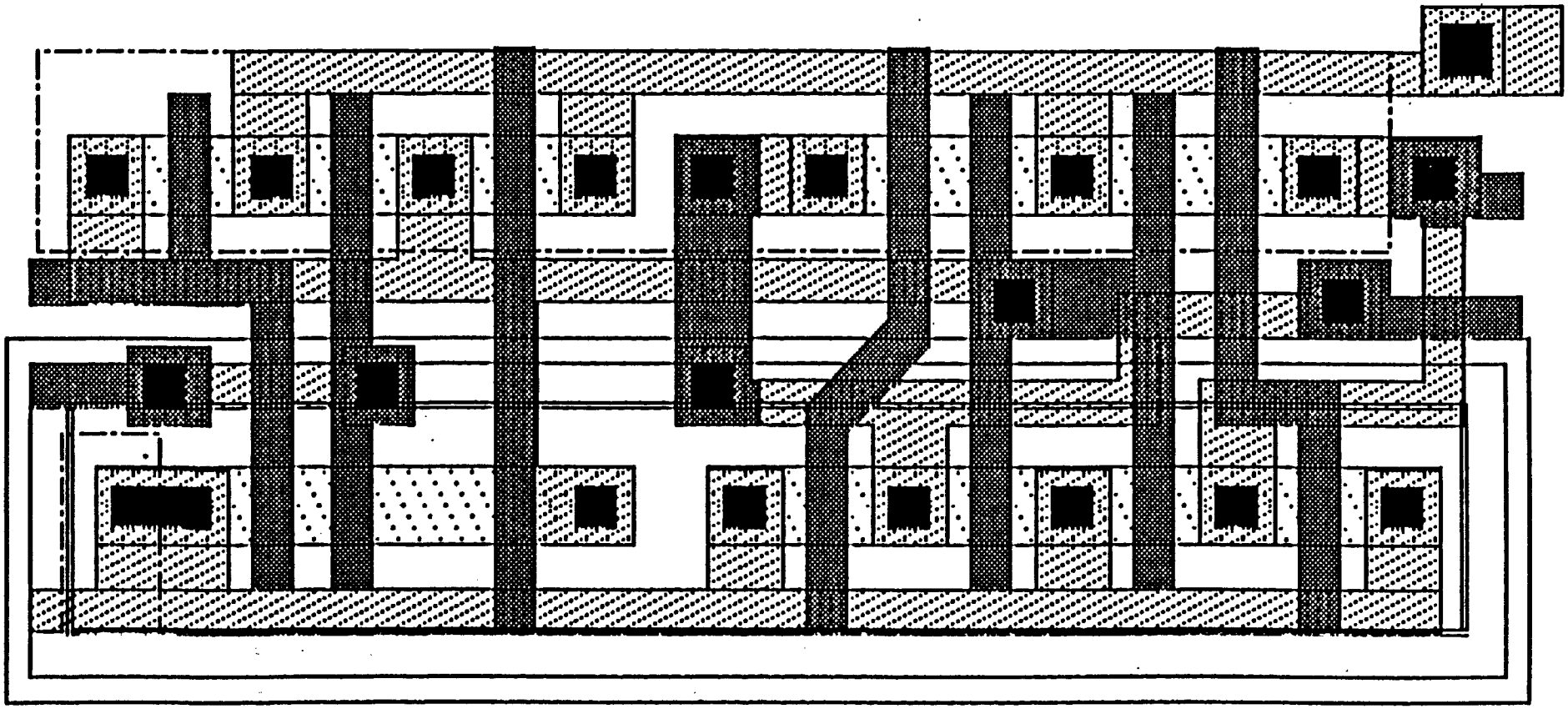
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



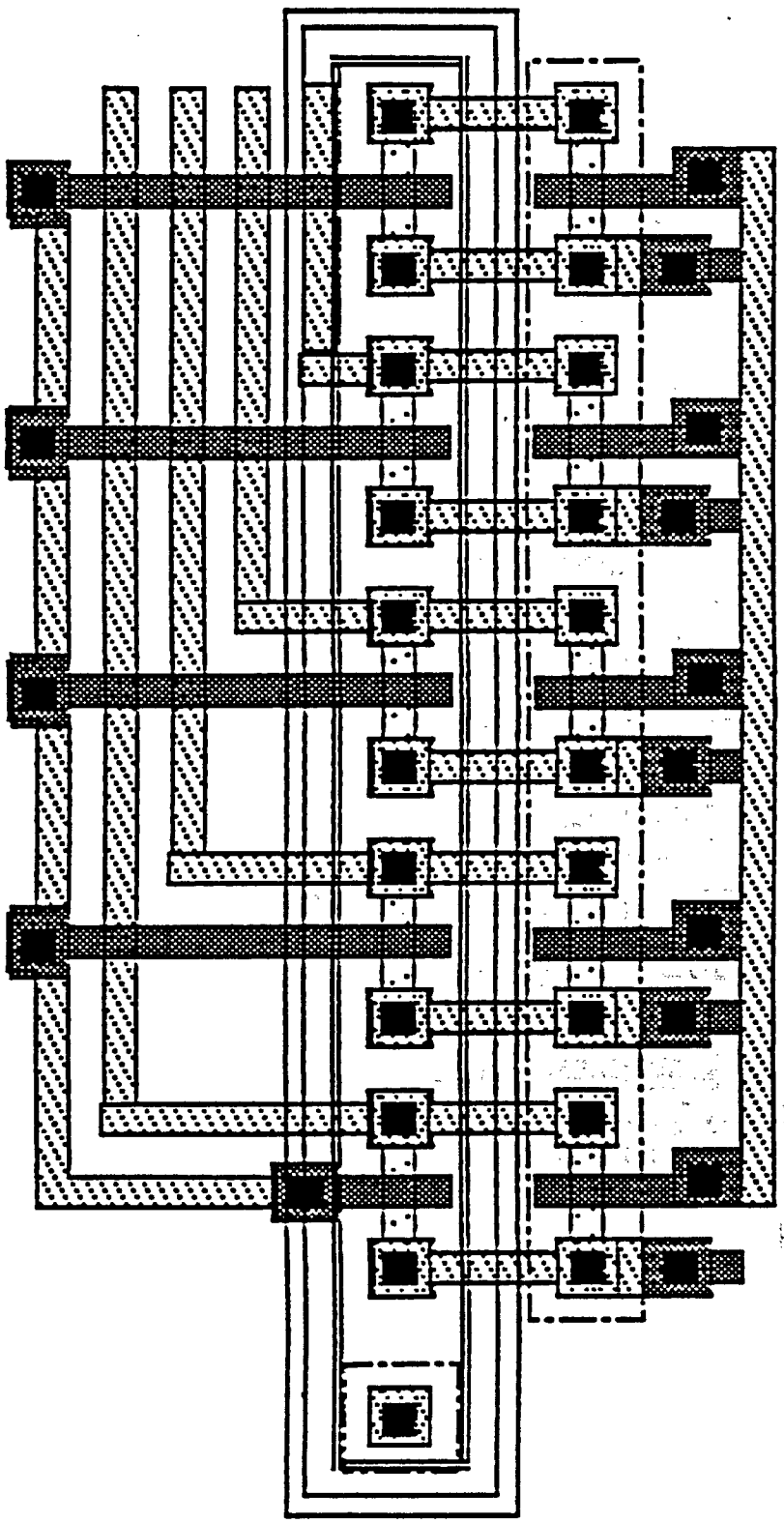
OPBUF2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



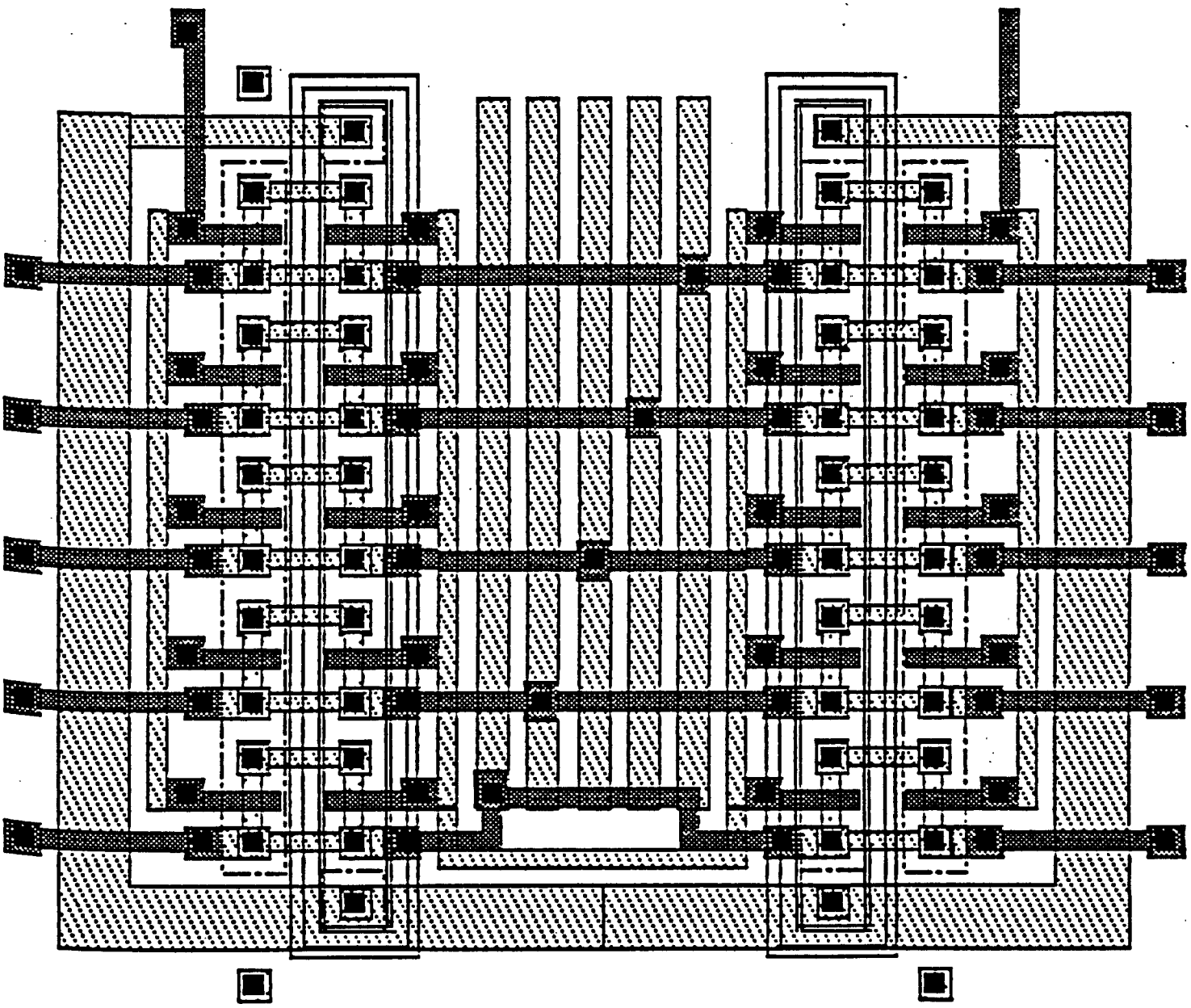
ROWDEC3



TRANSM2

101

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



TRANSM3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

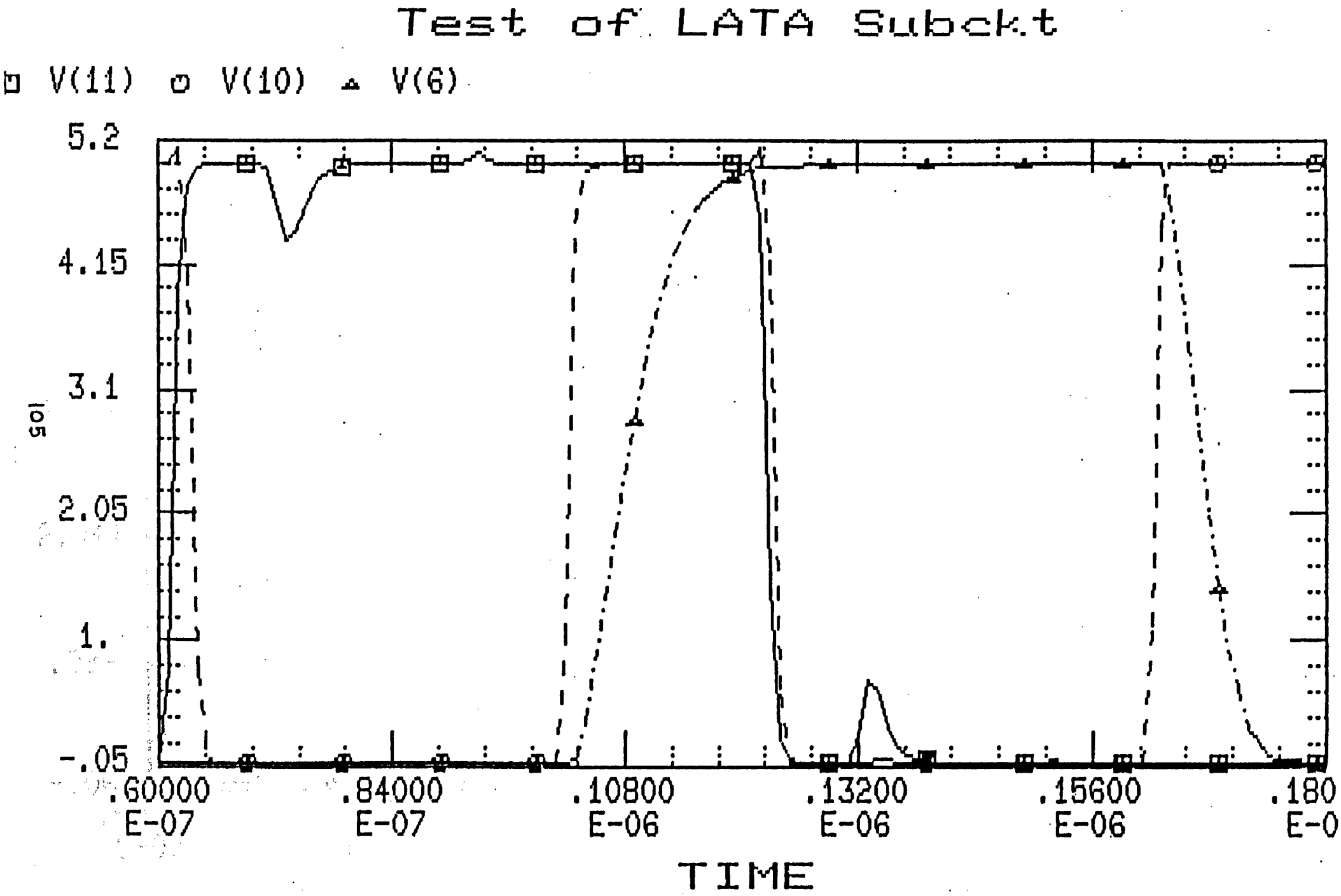
```
Test of LATA Subckt
*************************
* Start of Subcircuits
*************************
*--------------
* Dynamic Latch
*--------------
* 1=VCC 2=Din 3=Qout 4=ph1 5=ph1~ 6=ph2 7=ph2~ 9=internal
.SUBCKT LATA 1 2 3 4 5 6 7 9
M1 2  5  8 1 PMOSE W=5.4U L=3.0U AD=78P  AS=45P PD=40U PS=28U NRD=2.7 NRS=1.6
M2 9  8  1 1 PMOSE W=5.4U L=3.0U AD=39P  AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M3 9  7 10 1 PMOSE W=5.4U L=3.0U AD=42P  AS=42P PD=26U PS=26U NRD=1.4 NRS=1.4
M4 3 10  1 1 PMOSE W=5.4U L=3.0U AD=143P AS=42P PD=64U PS=26U NRD=4.9 NRS=1.4
M5 2  4  8 0 NMOSE W=3.0U L=3.0U AD=56P  AS=56P PD=40U PS=40U NRD=4.3 NRS=4.3
M6 9  8  0 0 NMOSE W=3.0U L=3.0U AD=36P  AS=36P PD=26U PS=26U NRD=2.6 NRS=2.6
M7 9  6 10 0 NMOSE W=3.0U L=3.0U AD=36P  AS=36P PD=26U PS=26U NRD=2.6 NRS=2.6
M8 3 10  0 0 NMOSE W=3.0U L=3.0U AD=35P  AS=92P PD=25U PS=64U NRD=2.1 NRS=8.8
.ENDS LATA
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*------------------
* Transmission gate
*------------------
* 1=VCC 2=INPUT 3=CTL~ 4=CTL 5=OUTPUT
.SUBCKT TRANSM2 1 2 3 4 5
M1 2 3 5 1 PMOSE W=3U L=3U AD=36P AS=36P PD=26U PS=26U NRD=2.6 NRS=2.6
M2 2 4 5 0 NMOSE W=3U L=3U AD=36P AS=36P PD=26U PS=26U NRD=2.6 NRS=2.6
.ENDS TRANSM2
*----------------------------------------
* load on output of latches for rowdec data
*----------------------------------------
* 1=input 2=output
.SUBCKT BLOADR 1 3
C1 1 0 61.1FF
R1 1 2 946
C2 2 0 63.6FF
R2 2 3 286
C3 3 0 92FF
.ENDS BLOADR
*********************
* Start of Circuits
*********************
VCC 1 0 DC 5

* 1=VCC 2=Din 3=Qout 4=ph1 5=ph1~ 6=ph2 7=ph2~ 9=internal
X1 1 11 6 8 7 10 9 12 LATA
X2 1 0 2 11 RATINV
X3 1 0 3 7  RATINV
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

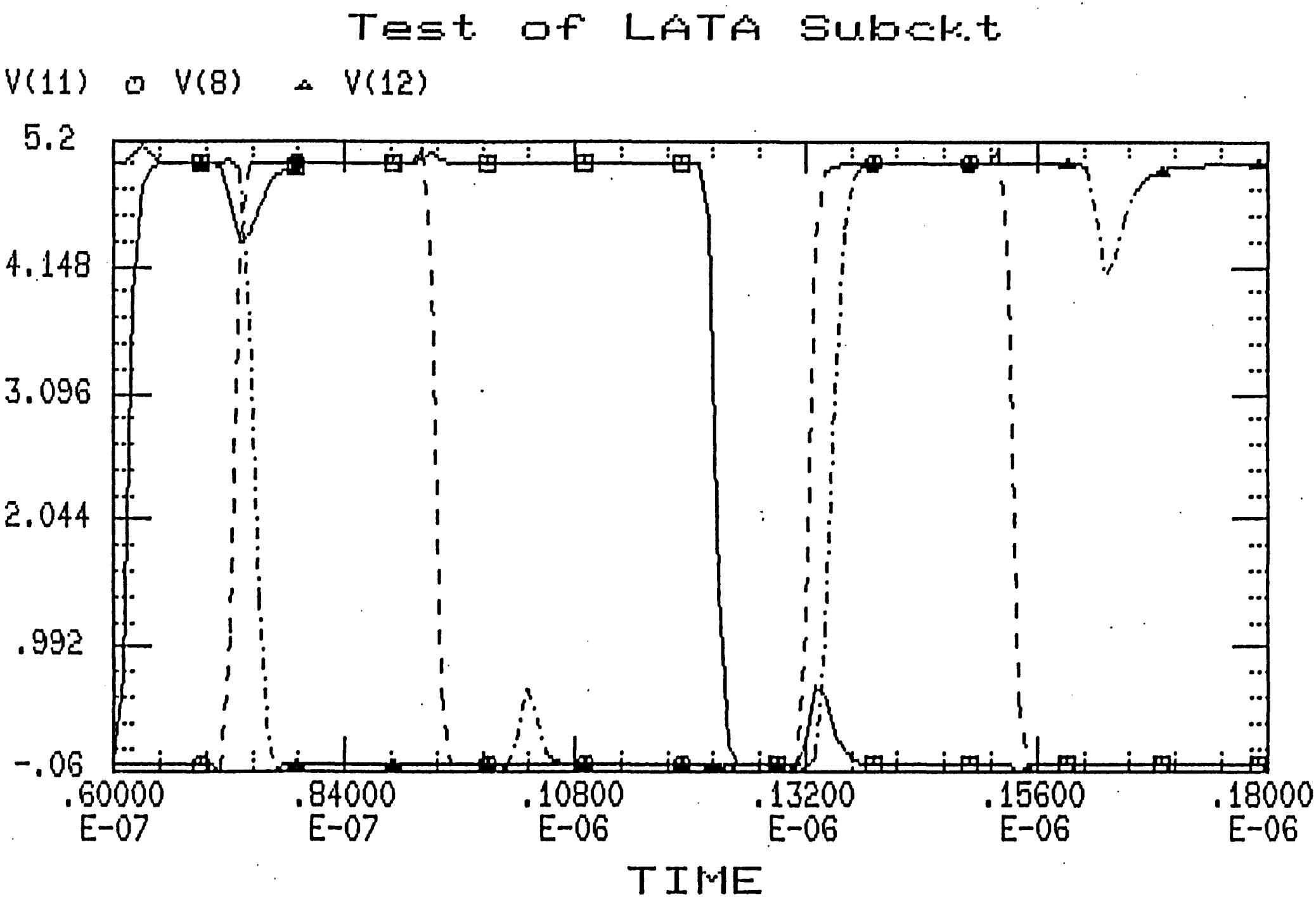
```
X4 1 0 7 8  RATINV
X5 1 0 4 9  RATINV
X6 1 0 9 10 RATINV
X7 6 5 BLOADR
*C1 12 0 23FF
* 1=VCC 2=INPUT 3=CTL~ 4=CTL 5=OUTPUT
*X8 1 12 1 0 13 TRANSM2
*IL2 13 0 DC 0
IL1 5 0 DC 0

VPH1 3 0 PULSE(0 5 10NS 0 0 20NS 60NS)
VPH2 4 0 PULSE(0 5 40NS 0 0 20NS 60NS)
VDAT 2 0 PULSE(5 0 60NS 0 0 60NS 120NS)

*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Start of Output Generation
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.TRAN 1.0NS 180NS 60NS
.PRINT TRAN V(11) V(8) V(12)
.PRINT TRAN V(11) V(10) V(6)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



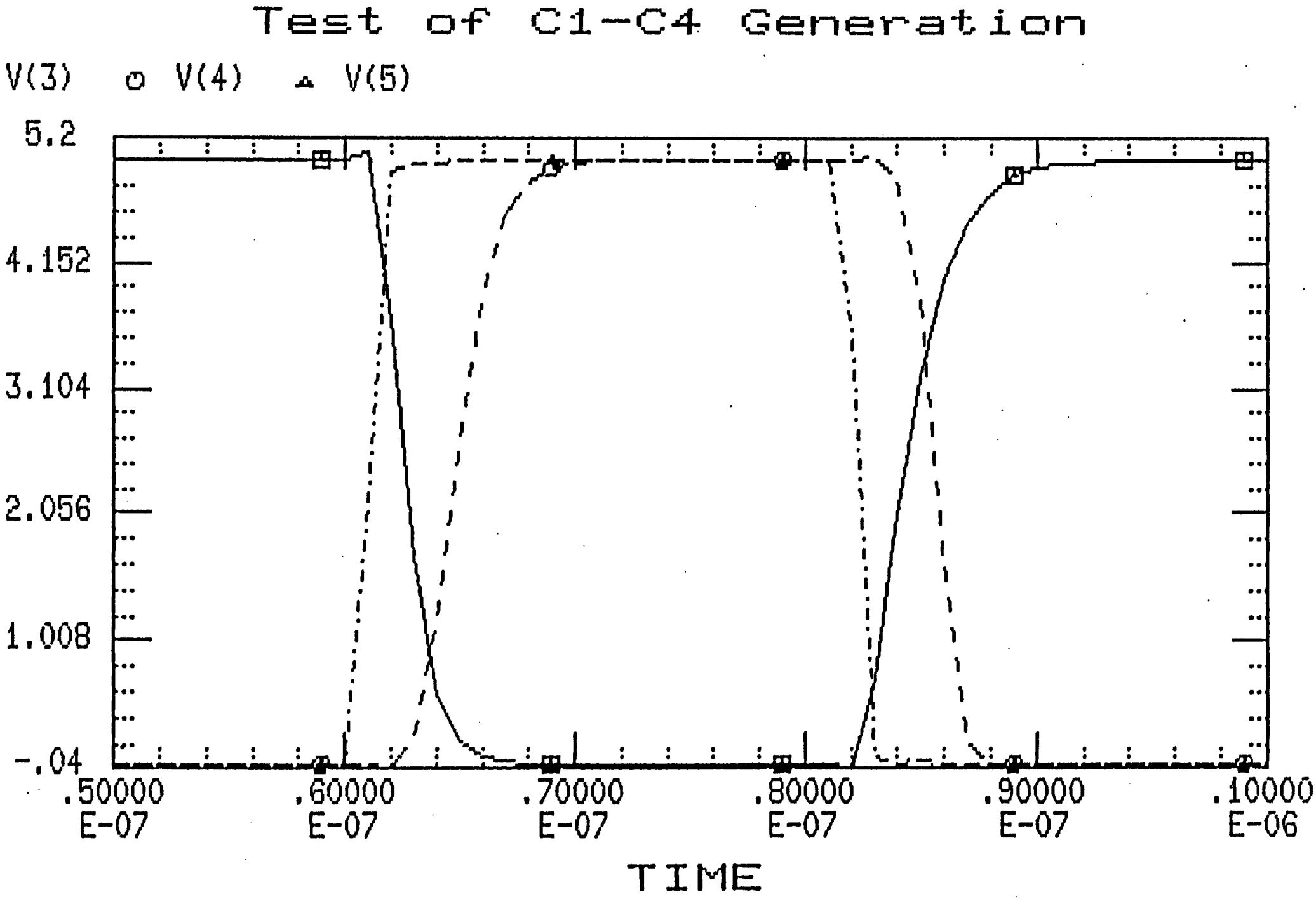
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of C1-C4 Generation
*++++++++++++++++++++++
* Start of Subcircuits
*++++++++++++++++++++++
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*---------------
* Column Decoder
*---------------
* 1=VCC 2=VIN1 3=VIN2 5=VOUT" 6=VOUT
.SUBCKT COLDEC 1 2 3 5 6
M1 5 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M2 5 3 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
M3 6 5 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
M4 5 3 4 0 NMOSE W=5.4U L=3.0U AD=39P AS=16P PD=25U PS=17U NRD=1.3 NRS=0.6
M5 4 2 0 0 NMOSE W=5.4U L=3.0U AD=16P AS=42P PD=17U PS=26U NRD=0.6 NRS=1.4
M6 6 5 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
.ENDS COLDEC
*++++++++++++++++++++
* Start of Circuits
*++++++++++++++++++++
VCC 1 0 DC 5

* 1=vcc 2=vin1 3=vin2 5=vout" 6=vout
X1 1 5 1 3 4 COLDEC
X2 1 0 2 5 RATINV
C1 4 0 113FF
C2 3 0 113FF

VPH1 2 0 PULSE(5 0 10NS 0 0 20NS 50NS)
*++++++++++++++++++++++++++++
* Start of Output Generation
*++++++++++++++++++++++++++++
.TRAN 1.0NS 100NS 50NS
.PRINT TRAN V(3) V(4) V(5)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

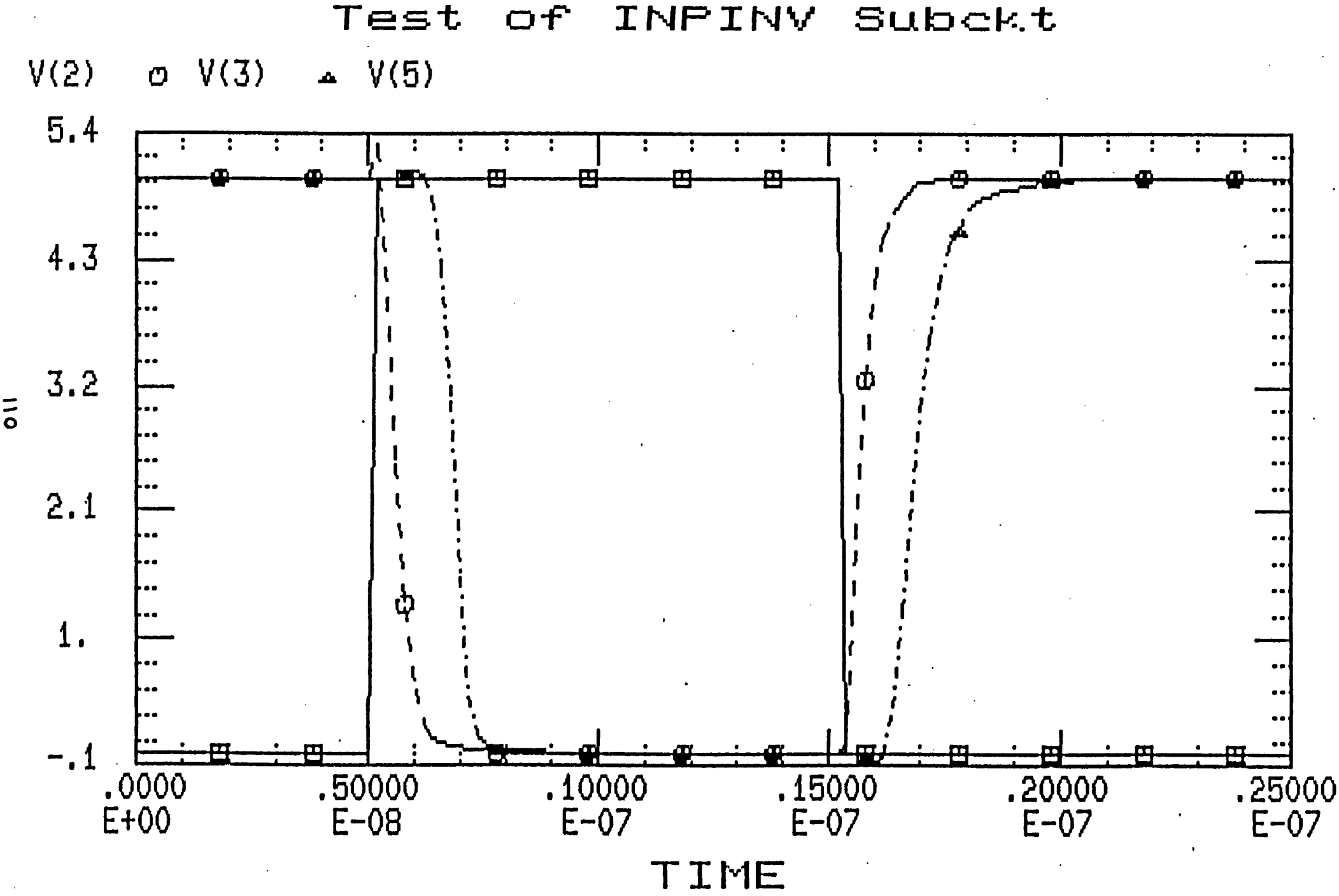
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of INPINV Subckt
*~~~~~~~~~~~~~~~~~~~~~
* Start of Subcircuits
*~~~~~~~~~~~~~~~~~~~~~
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*---------------
* Input Inverter
*---------------
* 1=VCC 2=INPUT 3=OUTPUT
.SUBCKT INPINV 1 2 3
M1 3 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
M2 3 2 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
.ENDS INPINV

*~~~~~~~~~~~~~~~~~~
* Start of Circuits
*~~~~~~~~~~~~~~~~~~
VCC 1 0 DC 5

X1 1 3 4 INPINV
X3 1 4 5 INPINV
X2 1 0 2 3 RATINV
X4 1 0 5 6 RATINV
CL 6 0 84FF

VIN 2 0 PULSE(0 5 5NS 0 0 10NS 25NS)

*~~~~~~~~~~~~~~~~~~~~~~~~~~
* Start of Output Generation
*~~~~~~~~~~~~~~~~~~~~~~~~~~
.TRAN 0.2NS 25NS
.PRINT TRAN V(2) V(3) V(5)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

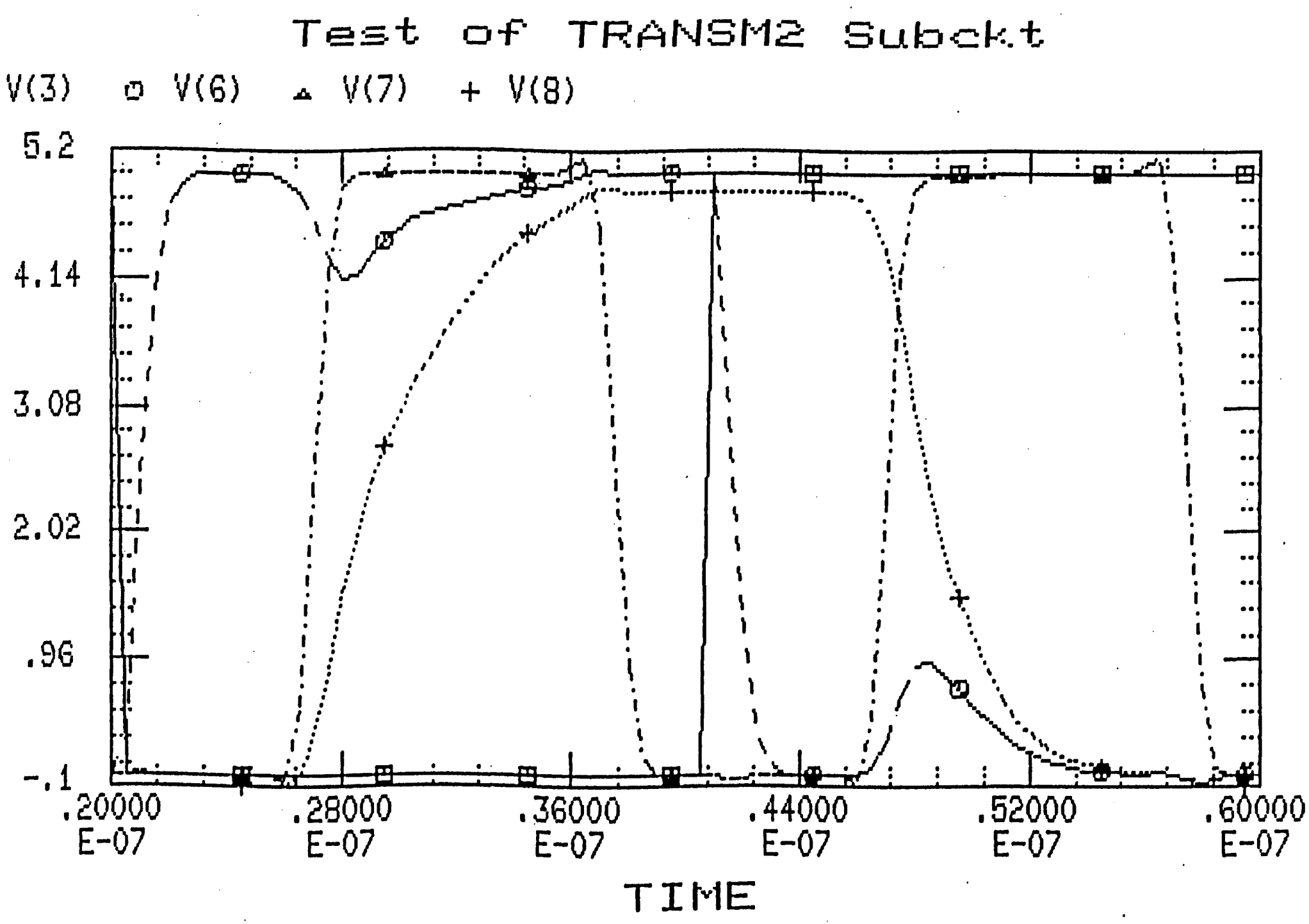
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of TRANSM2 Subckt
*~~~~~~~~~~~~~~~~~~~~~
* Start of Subcircuits
*~~~~~~~~~~~~~~~~~~~~~
*-----------------
* Transmission gate
*-----------------
* 1=VCC 2=INPUT 3=CTL~ 4=CTL 5=OUTPUT
.SUBCKT TRANSM2 1 2 3 4 5
M1 2 3 5 1 PMOSE W=3U L=3U AD=36P AS=36P PD=26U PS=26U NRD=2.6 NRS=2.6
M2 2 4 5 0 NMOSE W=3U L=3U AD=36P AS=36P PD=26U PS=26U NRD=2.6 NRS=2.6
.ENDS TRANSM2
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*~~~~~~~~~~~~~~~~~~
* Start of Circuits
*~~~~~~~~~~~~~~~~~~
VCC 1 0 DC 5

* 1=VCC 2=INPUT 3=CTL~ 4=CTL 5=OUTPUT
X1 1 6 5 7 8 TRANSM2
X6 1 8 1 0 4 TRANSM2
X2 1 0 2 5 RATINV
X3 1 0 3 6 RATINV
X4 1 0 5 7 RATINV
X5 1 0 8 9 RATINV
CL 8 0 23FF

VPH 2 0 PULSE(0 5 5NS 0 0 10NS 20NS)
VIN 3 0 PULSE(5 0 20NS 0 0 20NS 40NS)

*~~~~~~~~~~~~~~~~~~~~~~~~~~
* Start of Output Generation
*~~~~~~~~~~~~~~~~~~~~~~~~~~
.TRAN .5NS 60NS 20NS
.PRINT TRAN V(3) V(6) V(7) V(8)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



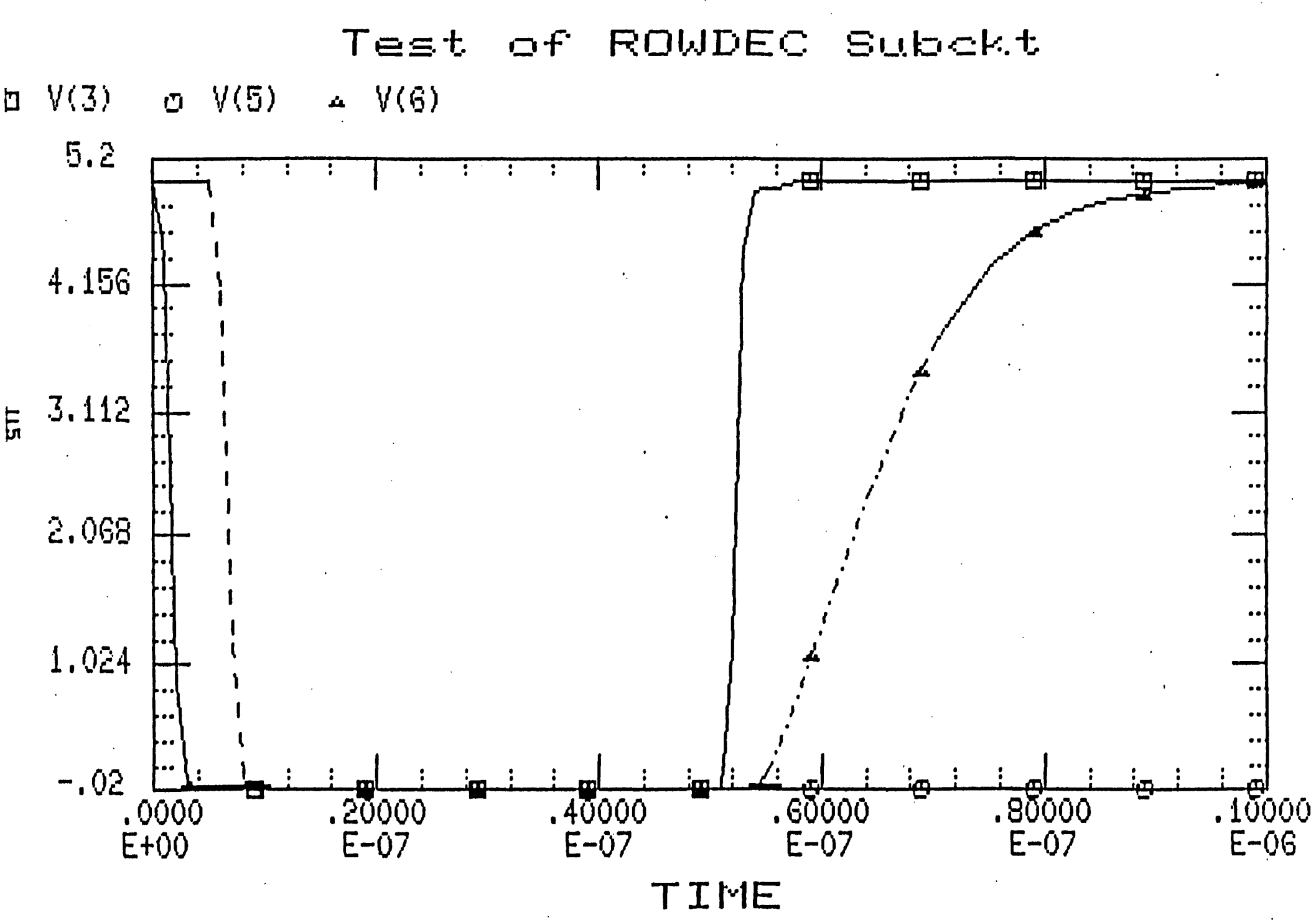
```
Test of ROWDEC Subckt
*~~~~~~~~~~~~~~~~~~~~
* Start of Subcircuits
*~~~~~~~~~~~~~~~~~~~~
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*------------
* Row Decoder
*------------
* 1=VCC 2=A 3=B 4=PRE~ 5=C 6=C~ 7=ROW2 8=ROW1
.SUBCKT ROWDEC 1 2 3 4 5 6 7 8
M1    9 2   1 1 PMOSE W=5.4U L=3.0U AD=39P AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M2    9 3   1 1 PMOSE W=5.4U L=3.0U AD=24P AS=39P PD=20U PS=25U NRD=0.8 NRS=1.3
M3    9 4   1 1 PMOSE W=5.4U L=3.0U AD=24P AS=39P PD=20U PS=25U NRD=0.8 NRS=1.3
M4   12 9   1 1 PMOSE W=5.4U L=3.0U AD=8P  AS=39P PD=14U PS=25U NRD=0.3 NRS=1.3
M5    7 5  12 1 PMOSE W=5.4U L=3.0U AD=39P AS=8P  PD=25U PS=14U NRD=1.3 NRS=0.3
M6   13 9   1 1 PMOSE W=5.4U L=3.0U AD=8P  AS=39P PD=14U PS=25U NRD=0.3 NRS=1.3
M7    8 6  13 1 PMOSE W=5.4U L=3.0U AD=39P AS=8P  PD=25U PS=14U NRD=1.3 NRS=0.3
M8    9 4  10 0 NMOSE W=5.4U L=3.0U AD=39P AS=32P PD=25U PS=23U NRD=1.3 NRS=1.1
M9   10 3  11 0 NMOSE W=5.4U L=3.0U AD=16P AS=8P  PD=17U PS=14U NRD=0.6 NRS=0.3
M10  11 2   0 0 NMOSE W=5.4U L=3.0U AD=8P  AS=39P PD=14U PS=25U NRD=0.3 NRS=1.3
M11   7 5   0 0 NMOSE W=5.4U L=3.0U AD=24P AS=39P PD=20U PS=25U NRD=0.8 NRS=1.3
M12   7 9   0 0 NMOSE W=5.4U L=3.0U AD=24P AS=39P PD=20U PS=25U NRD=0.8 NRS=1.3
M13   8 6   0 0 NMOSE W=5.4U L=3.0U AD=24P AS=39P PD=20U PS=25U NRD=0.8 NRS=1.3
M14   8 9   0 0 NMOSE W=5.4U L=3.0U AD=24P AS=39P PD=20U PS=25U NRD=0.8 NRS=1.3
.ENDS ROWDEC
*---------------------------------
* row 1 subcircuit for binrom test
*---------------------------------
* 1=input 2=output
.SUBCKT ROW1 1 2
R1 1 2 449
C1 2 0 66.6FF
.ENDS ROW1
*---------------
* Input Inverter
*---------------
* 1=VCC 2=INPUT 3=OUTPUT
.SUBCKT INPINV 1 2 3
M1 3 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
M2 3 2 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
.ENDS INPINV
*-------------------------
* Input Inverter with load
*-------------------------
* 1=VCC 2=INPUT 3=OUTPUT
.SUBCKT TOT_INV 1 2 3
X1 1 2 3 INPINV
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
CL 3 0 70FF
.ENDS TOT_INV
*~~~~~~~~~~~~~~~~~~
* Start of Circuits
*~~~~~~~~~~~~~~~~~~
VCC 1 0 DC 5

* 1=VCC 2=A 3=B 4=PRE~ 5=C 6=C~ 7=ROW2 8=ROW1
X1 1 1 1 3 1 5 7 6 ROWDEC
X4 1 0 4 3 RATINV
X3 1 2 5 TOT_INV
XR1 6  8  ROW1
XR2 8  9  ROW1
XR3 9  10 ROW1
XR4 10 11 ROW1
IL2 11 0 DC 0
IL1 7  0 DC 0

VC  2 0 PULSE(0 5 5NS 0 0 95NS 100NS)
VPR 4 0 PULSE(0 5 0 0 0 50NS 100NS)

*~~~~~~~~~~~~~~~~~~~~~~~~~~
* Start of Output Generation
*~~~~~~~~~~~~~~~~~~~~~~~~~~
.TRAN 1.0NS 100NS
.PRINT TRAN  V(3) V(5) V(6)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

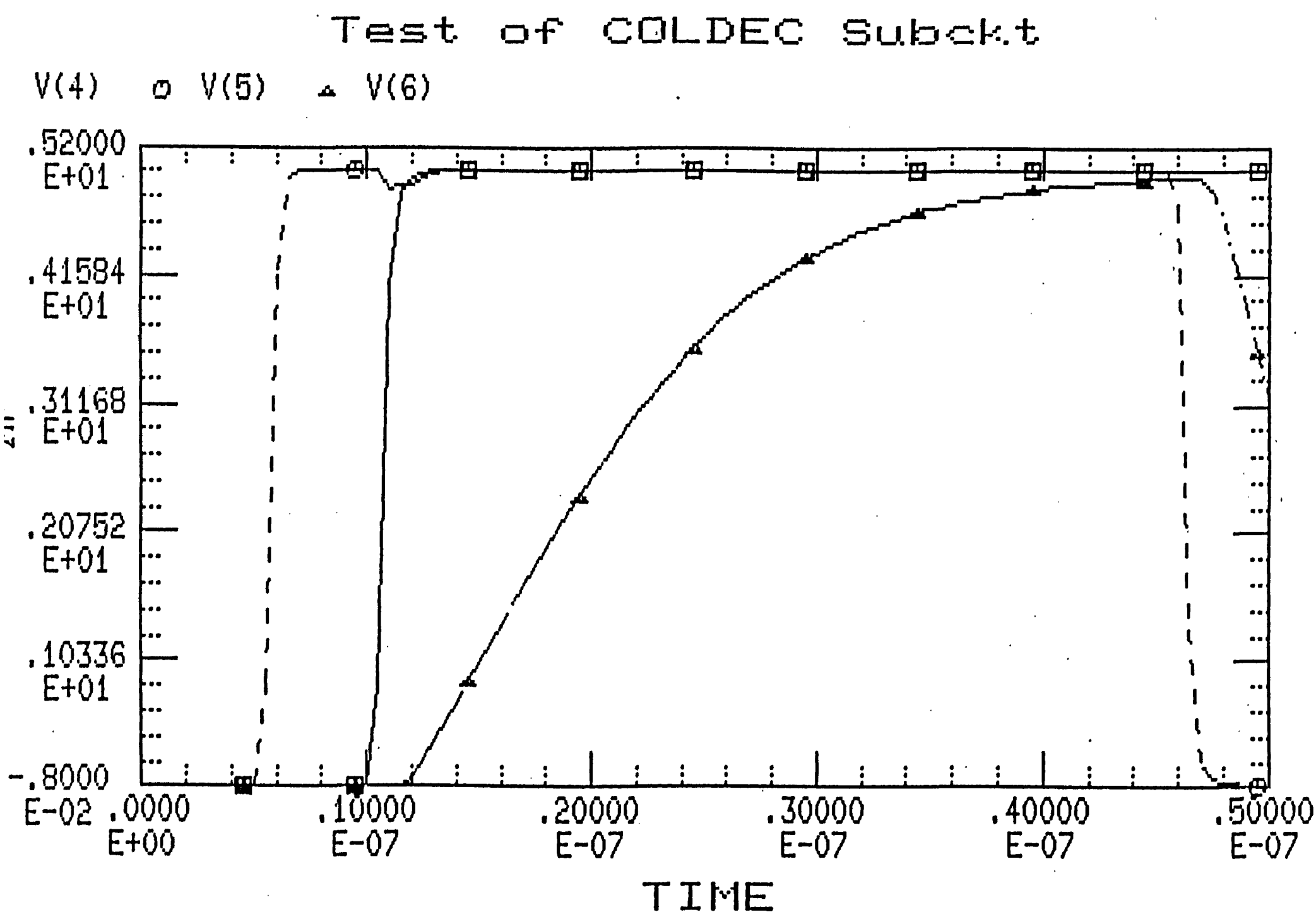
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of COLDEC Subckt
*~~~~~~~~~~~~~~~~~~~~~
* Start of Subcircuits
*~~~~~~~~~~~~~~~~~~~~~
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*---------------
* Column Decoder
*---------------
* 1=VCC 2=VIN1 3=VIN2 5=VOUT~ 6=VOUT
.SUBCKT COLDEC 1 2 3 5 6
M1 5 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M2 5 3 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
M3 6 5 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
M4 5 3 4 0 NMOSE W=5.4U L=3.0U AD=39P AS=16P PD=25U PS=17U NRD=1.3 NRS=0.6
M5 4 2 0 0 NMOSE W=5.4U L=3.0U AD=16P AS=42P PD=17U PS=26U NRD=0.6 NRS=1.4
M6 6 5 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
.ENDS COLDEC
*---------------------------------
* col 1 subcircuit for binrom test
*---------------------------------
* 1=input/output
.SUBCKT COL1 1
R1 1 2 33
C1 1 0 33.3FF
C2 1 0 33.3FF
R2 2 3 410
R3 2 4 410
C3 3 0 65.4FF
C4 4 0 65.4FF
.ENDS COL1
*~~~~~~~~~~~~~~~~~~~
* Start of Circuits
*~~~~~~~~~~~~~~~~~~~
VCC 1 0 DC 5

X1 1 4 5 7 6 COLDEC
X2 1 0 2 4 RATINV
X3 1 0 3 5 RATINV
X4 1 0 6 8 RATINV
XC1 6 COL1
XC2 6 COL1
XC3 6 COL1
XC4 6 COL1
ILOAD 7 0 DC 0

V1 2 0 PULSE(5 0 10NS 0 0 40NS 50NS)
V2 3 0 PULSE(5 0  5NS 0 0 40NS 50NS)
*~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Start of Output Generation
*~~~~~~~~~~~~~~~~~~~~~~~~~~~
.TRAN 0.5NS 50NS
.PRINT TRAN V(4) V(5) V(6)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



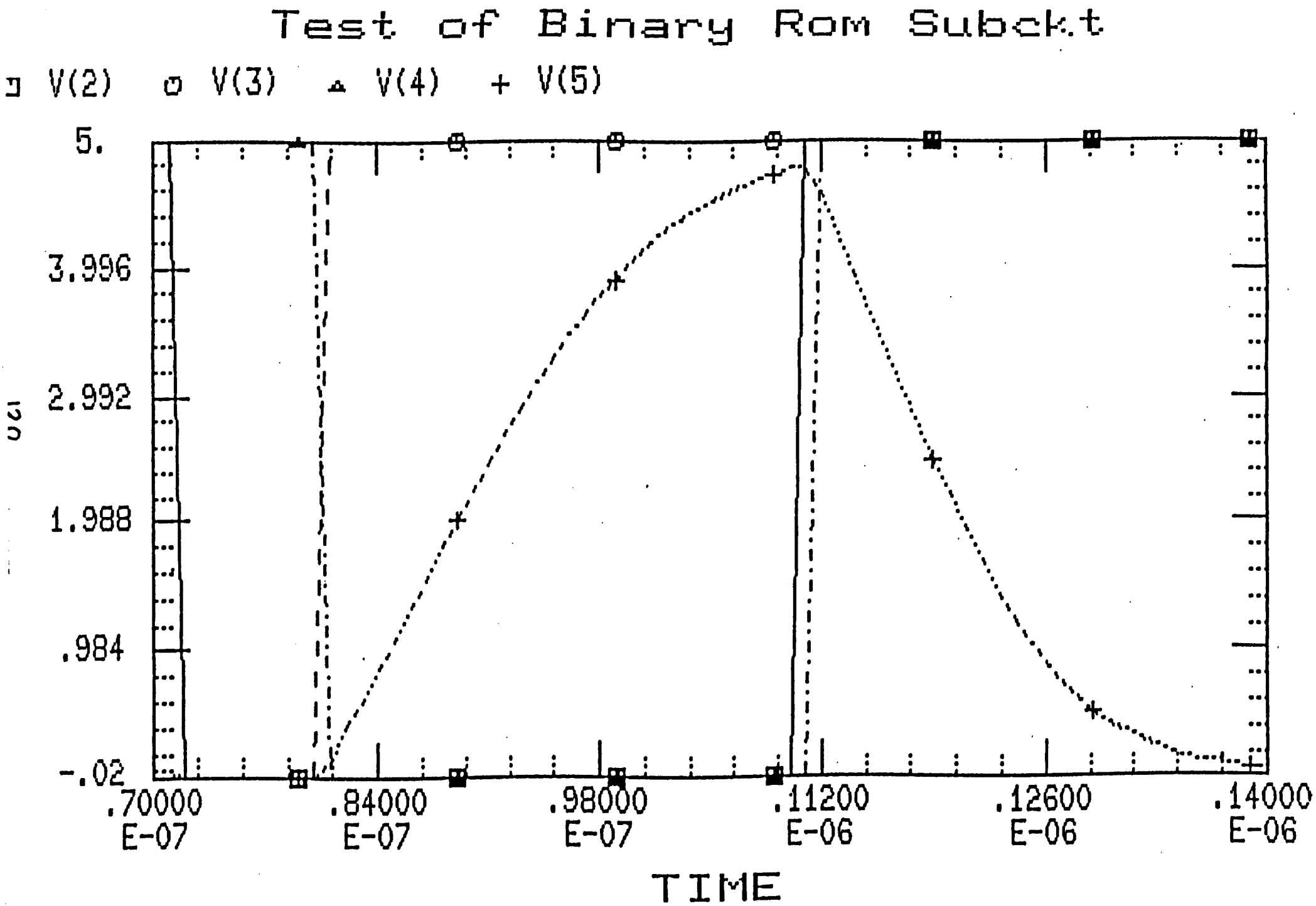
```
Test of Binary Rom Subckt
***********************
* Start of Subcircuits
***********************
*---------------------------------
* row 1 subcircuit for binrom test
*---------------------------------
* 1=input 2=output
.SUBCKT ROW1 1 2
R1 1 2 449
C1 2 0 66.6FF
.ENDS ROW1
*---------------------------------
* row 2 subcircuit for binrom test
*---------------------------------
* 1=input 2=output
.SUBCKT ROW2 1 2
R1 1 2 225
C1 2 0 33.3FF
R2 2 3 225
C2 3 0 33:3FF
.ENDS ROW2
*---------------------------------
* col 1 subcircuit for binrom test
*---------------------------------
* 1=input/output
.SUBCKT COL1 1
R1 1 2 33
C1 1 0 33.3FF
C2 1 0 33.3FF
R2 2 3 410
R3 2 4 410
C3 3 0 65.4FF
C4 4 0 65.4FF
.ENDS COL1
*---------------------------------
* col 2 subcircuit for binrom test
*---------------------------------
* 1=input 4=output
.SUBCKT COL2 1 4
R1 1 2 33
C1 1 0 33.3FF
C2 1 0 33.3FF
R2 2 3 410
R3 2 4 205
C3 3 0 65.4FF
C4 4 0 32FF
R4 4 5 205
C5 5 0 32FF
.ENDS COL2
*---------------------------------
* bit 1 subcircuit for binrom test
*---------------------------------
* 1=input/output
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
.SUBCKT BIT1 1
R1 1 2 8.3
C1 2 0 146FF
C2 1 0 25.6FF
.ENDS BIT1

*********************
* Start of Circuits
*********************
VCC 1 0 DC 5

XR1 2 6 ROW1
XR2 6 7 ROW1
XR3 7 8 ROW1
XR4 8 9 ROW2
XC1 3 COL1
XC2 3 COL1
XC3 3 COL1
XC4 3 13 COL2
XB1 10 BIT1
XB2 11 BIT1
XB3 12 BIT1
XB4 5  BIT1
M1  5 13 15 0 NMOSE W=5.4U L=3U AD=13P AS=13P PD=26U PS=26U NRD=0.5 NRS=0.5
M2 15  9  0 0 NMOSE W=5.4U L=3U AD=13P AS=13P PD=26U PS=26U NRD=0.5 NRS=0.5
M3 14  4  1 1 PMOSE W=7.2U L=3U AD=52P AS=48P PD=29U PS=28U NRD=1.0 NRS=1.0
R1 10 5 396
R2 11 5 396
R3 12 5 396
R4 14 5 110
C1 5 0 35FF

VPR 4 0 PULSE(5 0 10NS 0 0 30NS 70NS)
VCS 3 0 PULSE(0 5 10NS 0 0 60NS 70NS)
VRS 2 0 PULSE(0 5 40NS 0 0 30NS 70NS)

*****************************
* Start of Output Generation
*****************************
.TRAN 1.0NS 140NS 70NS
.PRINT TRAN V(2) V(3) V(4) V(5)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Appendix III

Improved Binary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Pin #	I/O	Function
1	I	B0
2	I	Phi 2
3	I	Phi 1
4	I	Phi 1 bar
5	I	Phi 2 bar
6	O	B0
7	O	B1
8	O	B2
9	O	B3
10	O	B4
11	I	A4
12	I	A3
13	I	A2
14	I	A1
15	I	A0
16	I	Vdd
17	I	B4
18	I	B3
19	I	B2
20	I	B1
21	I	B0
22	I	Vss

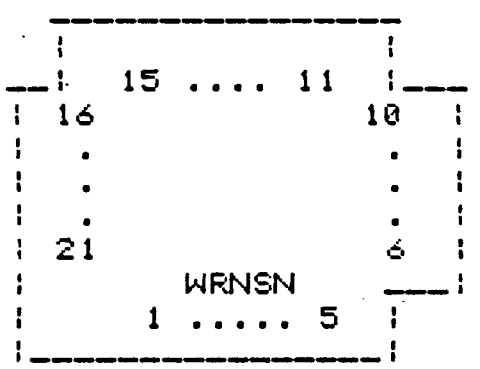
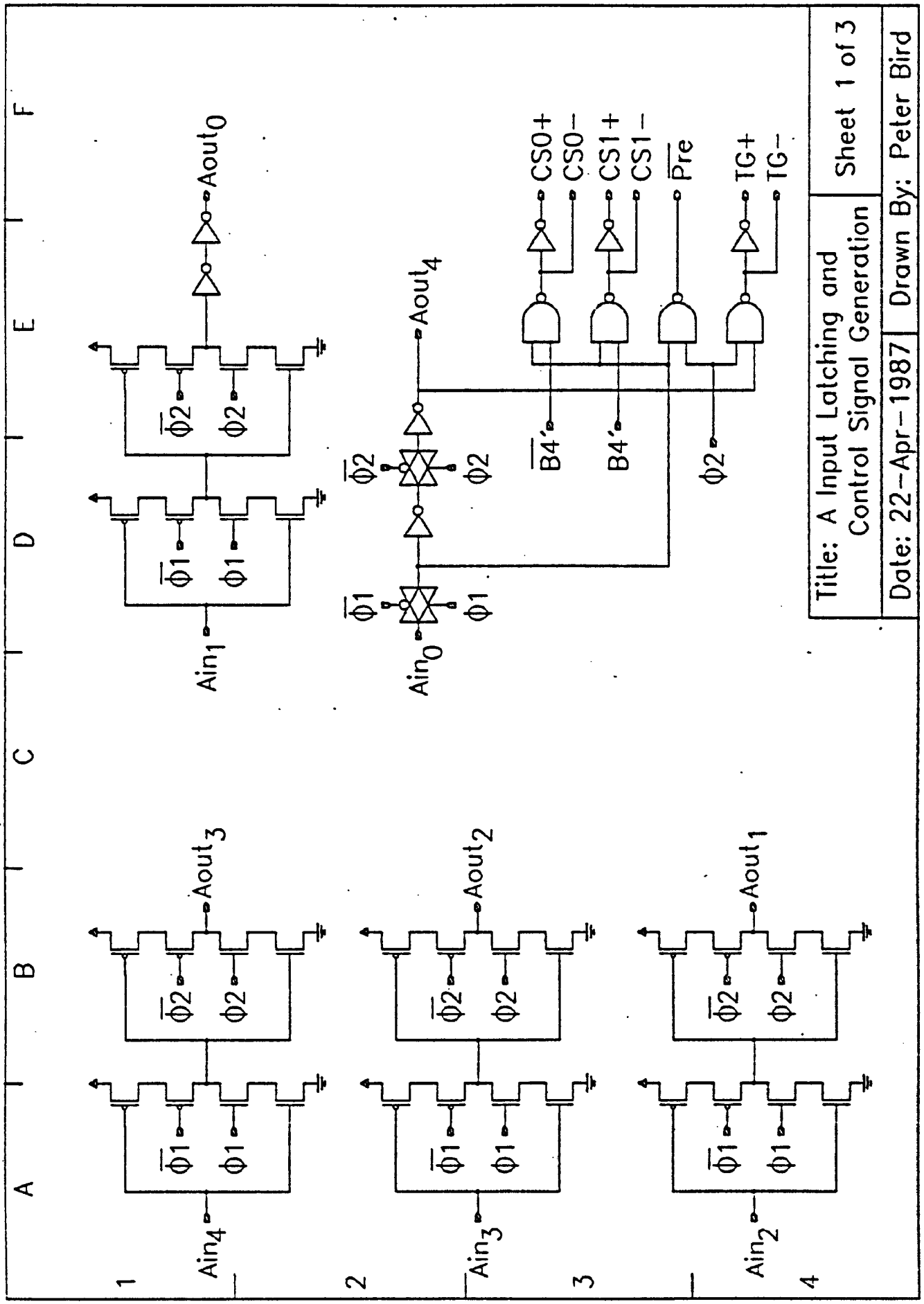


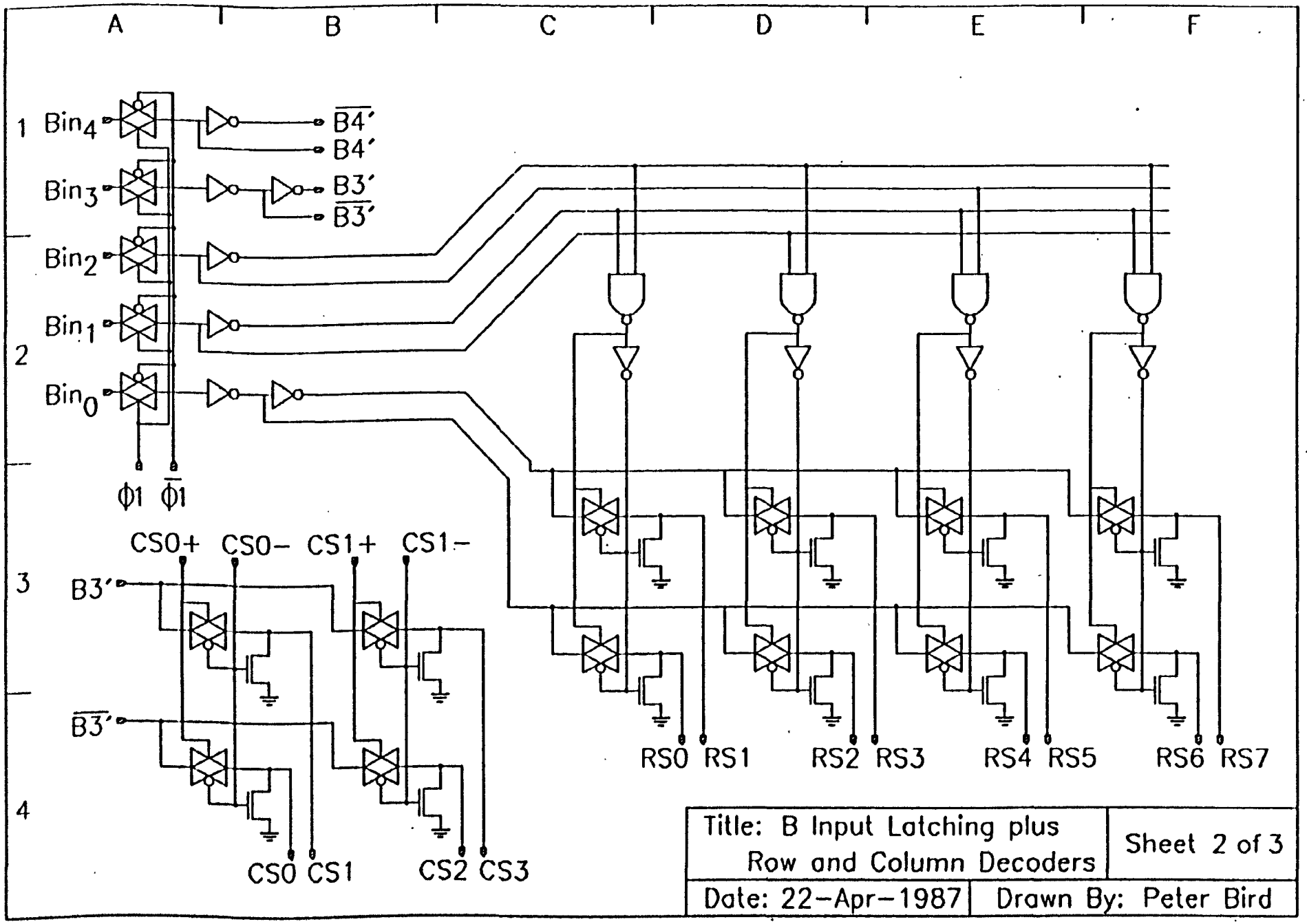
Figure A.4. Pinouts for the Improved Binary Design

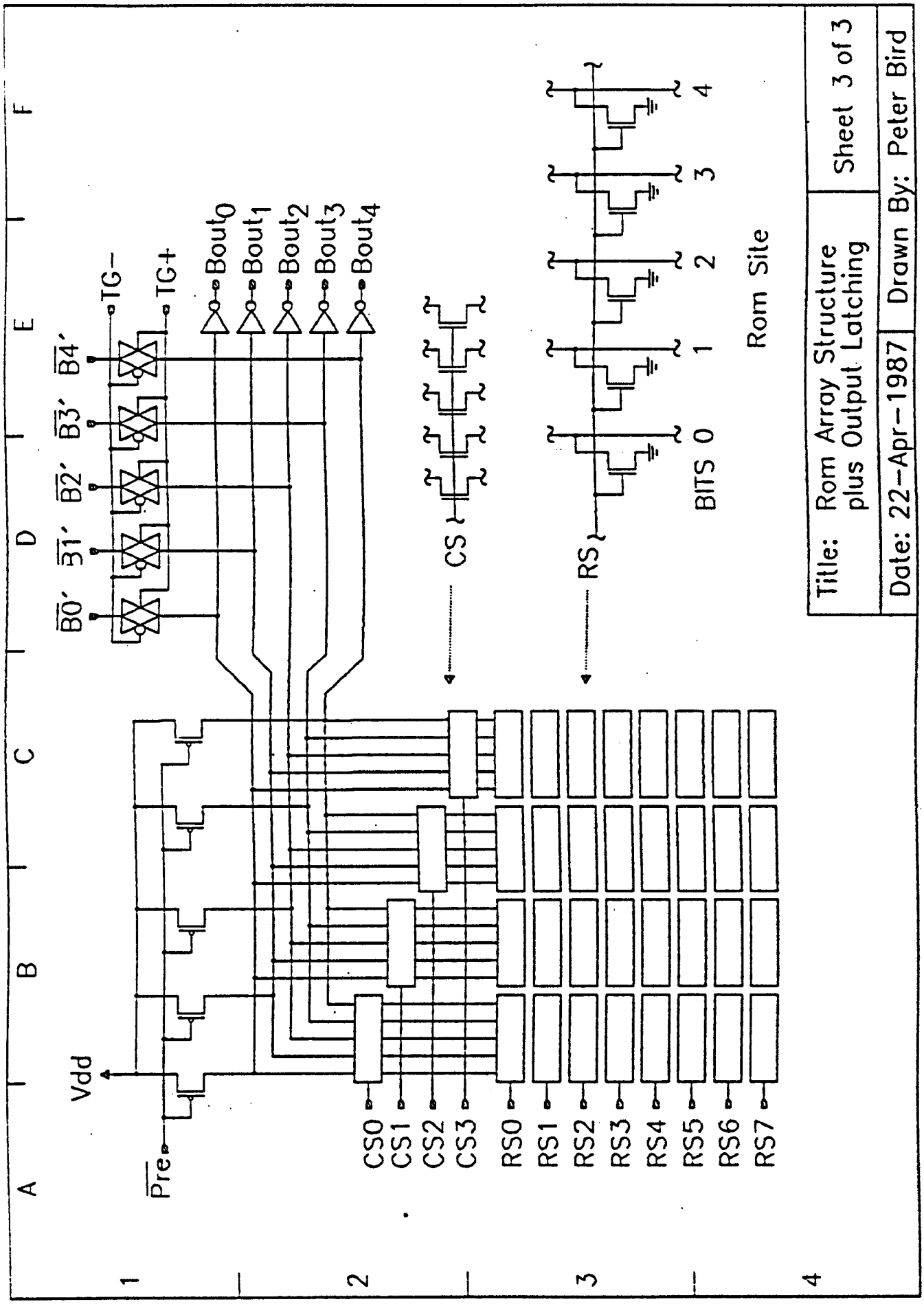
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



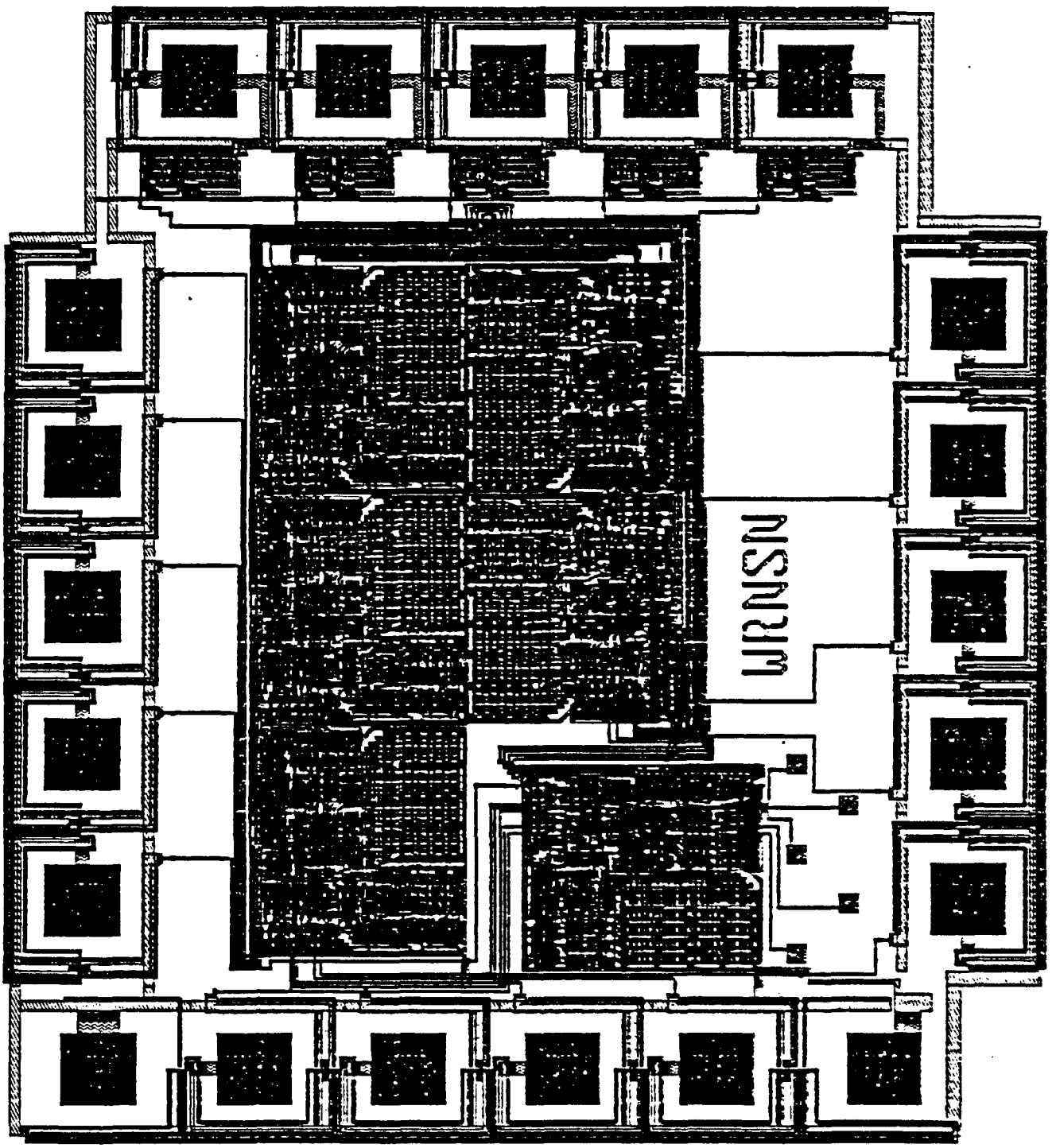
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

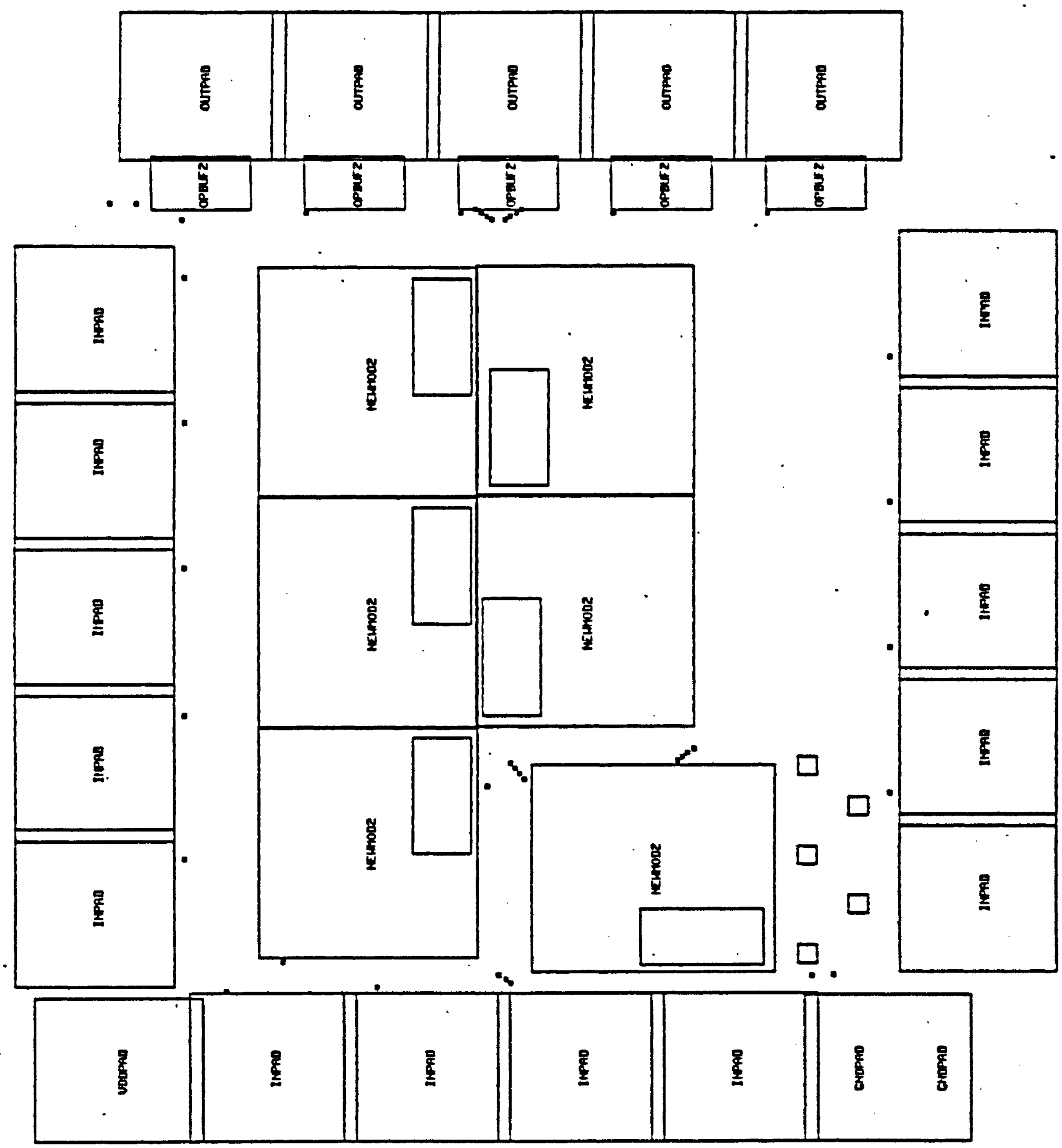




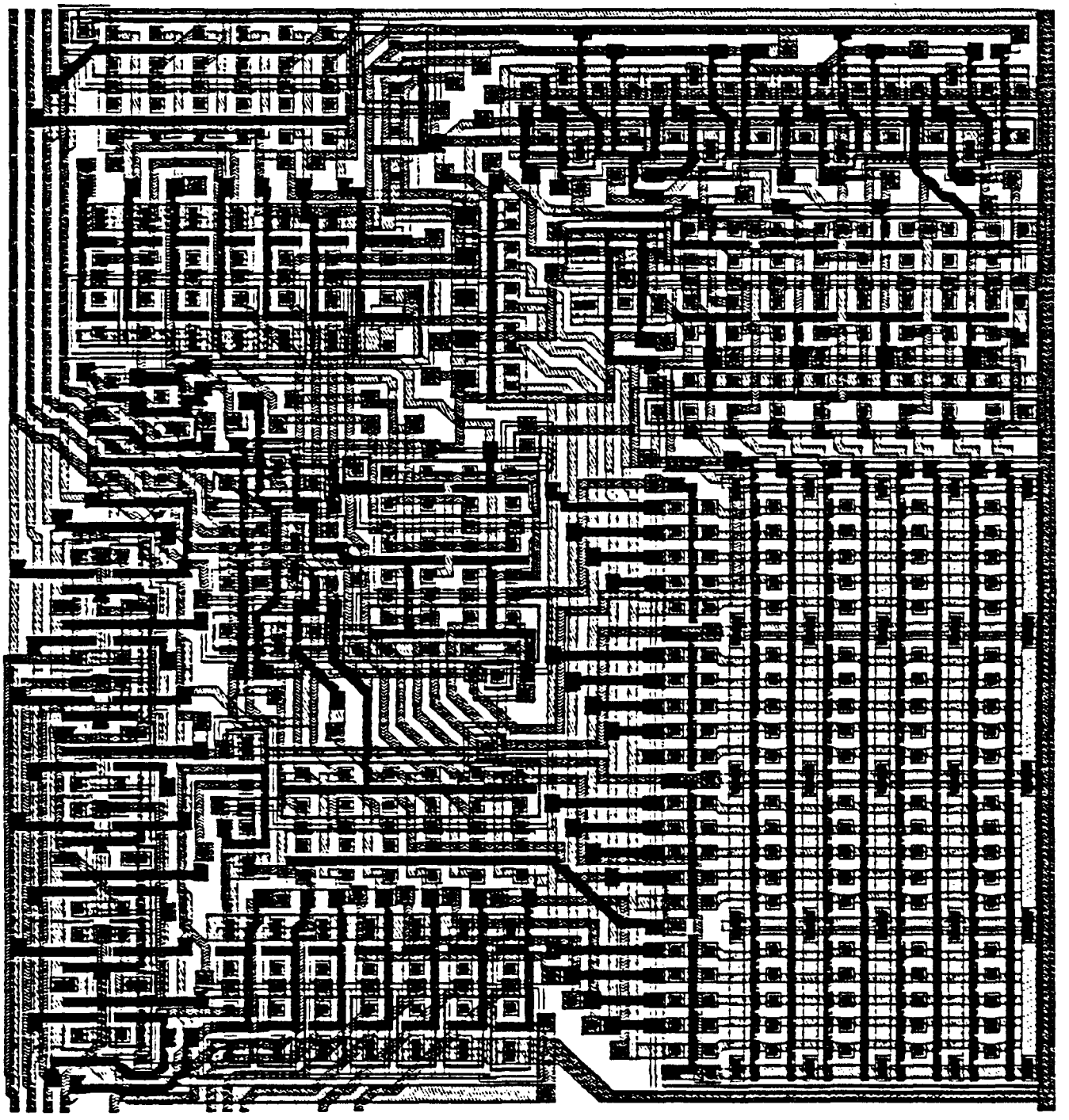
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



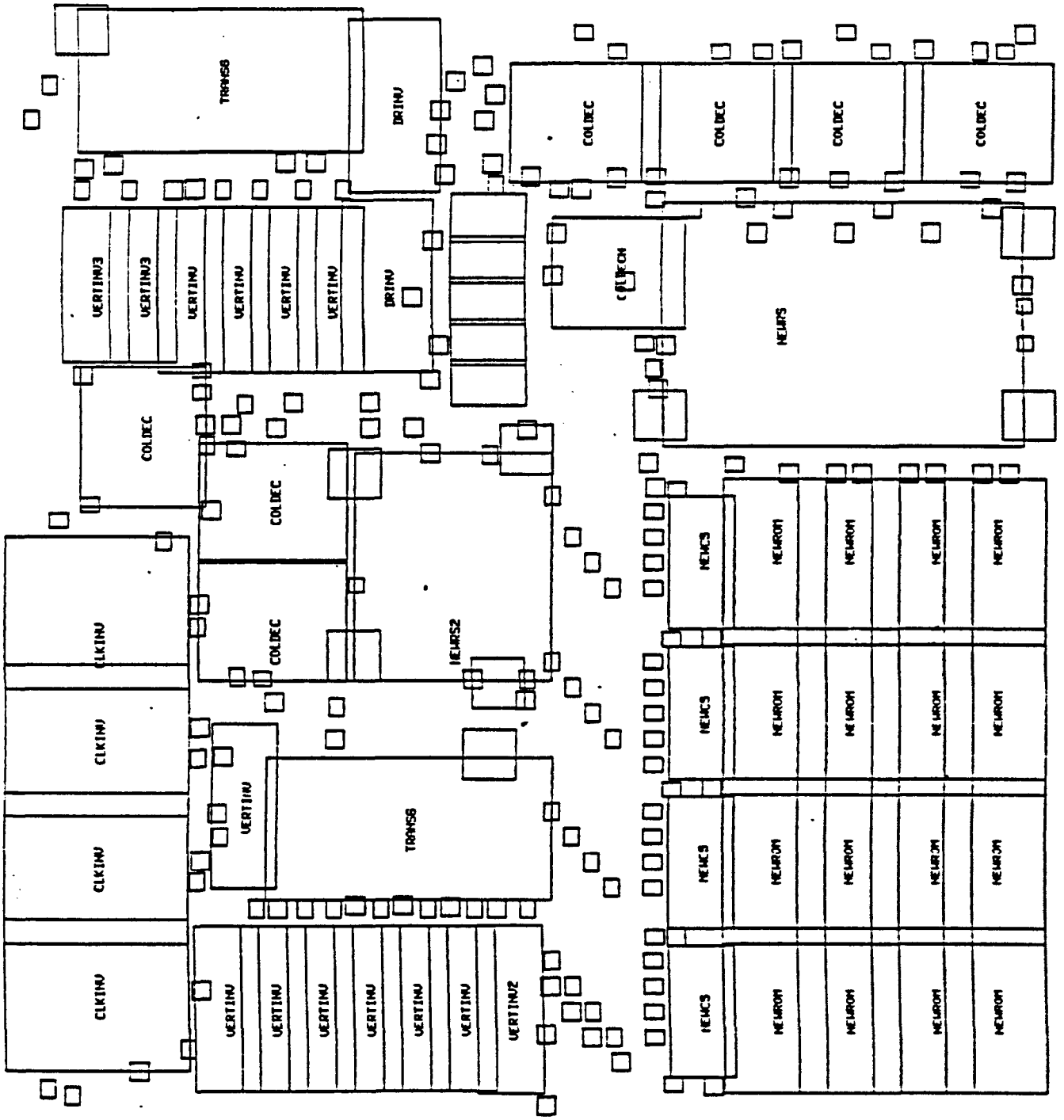
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



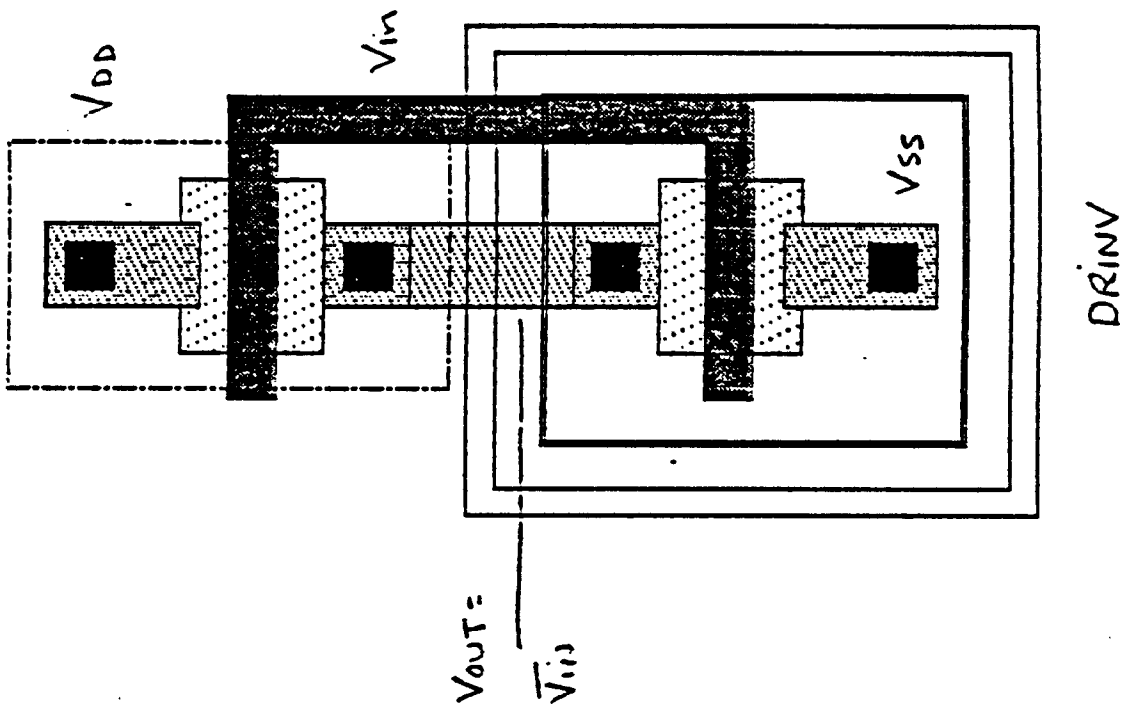
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



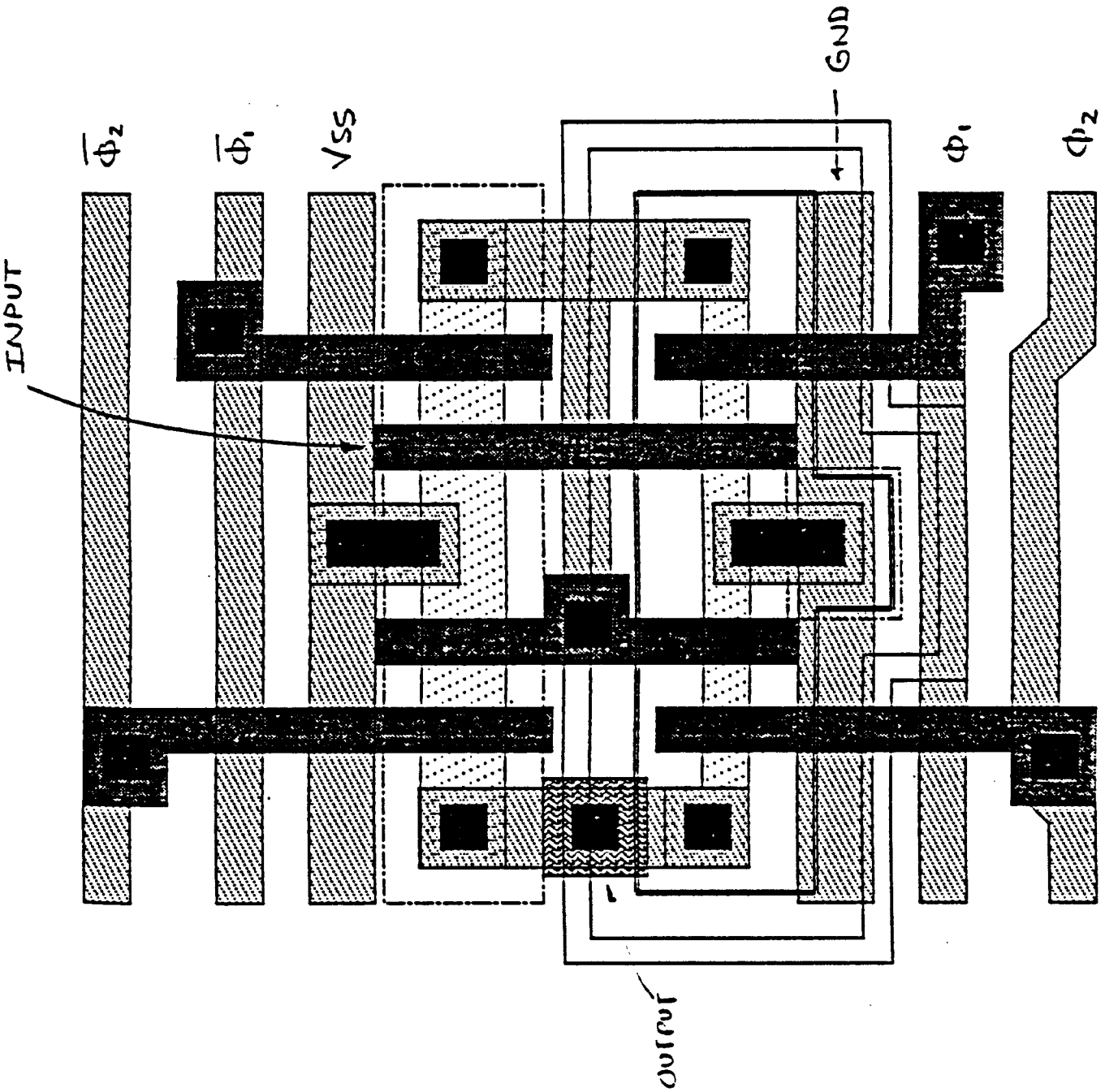
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

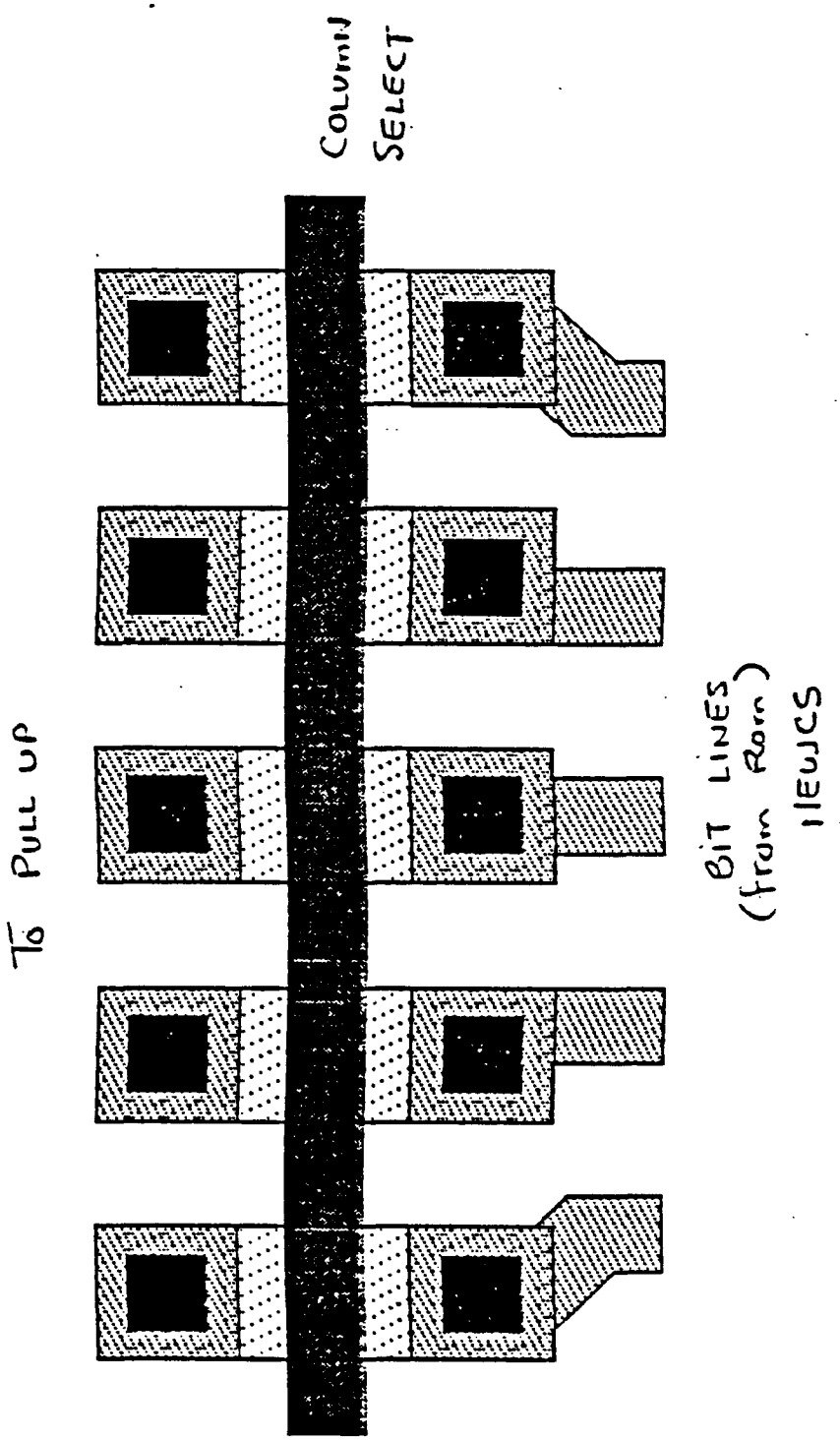


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

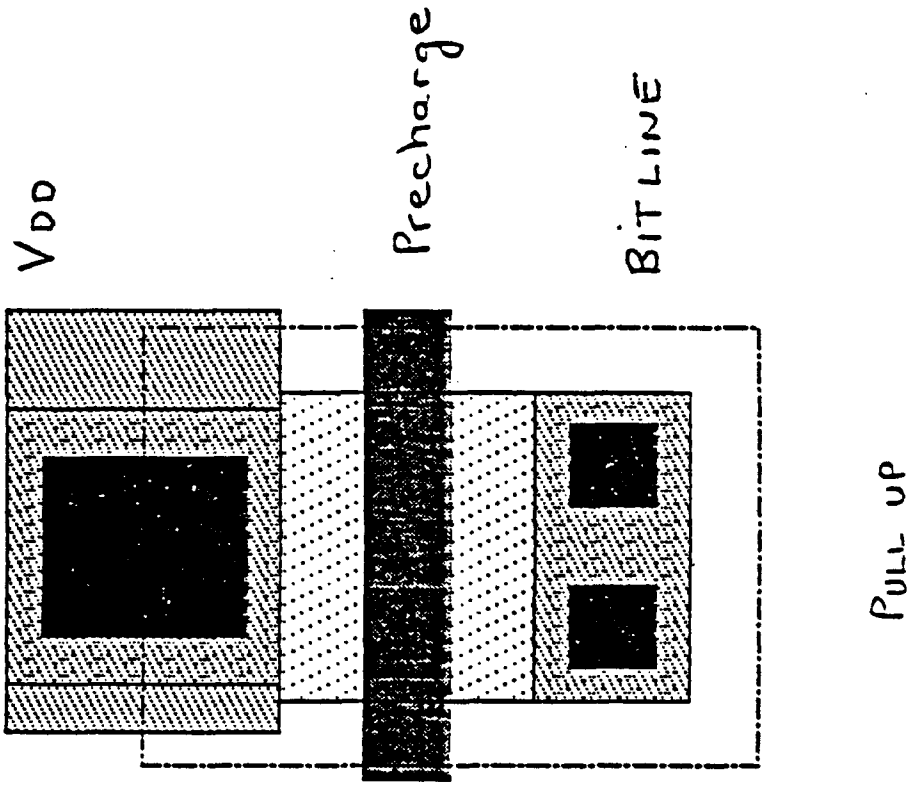


CLKINV

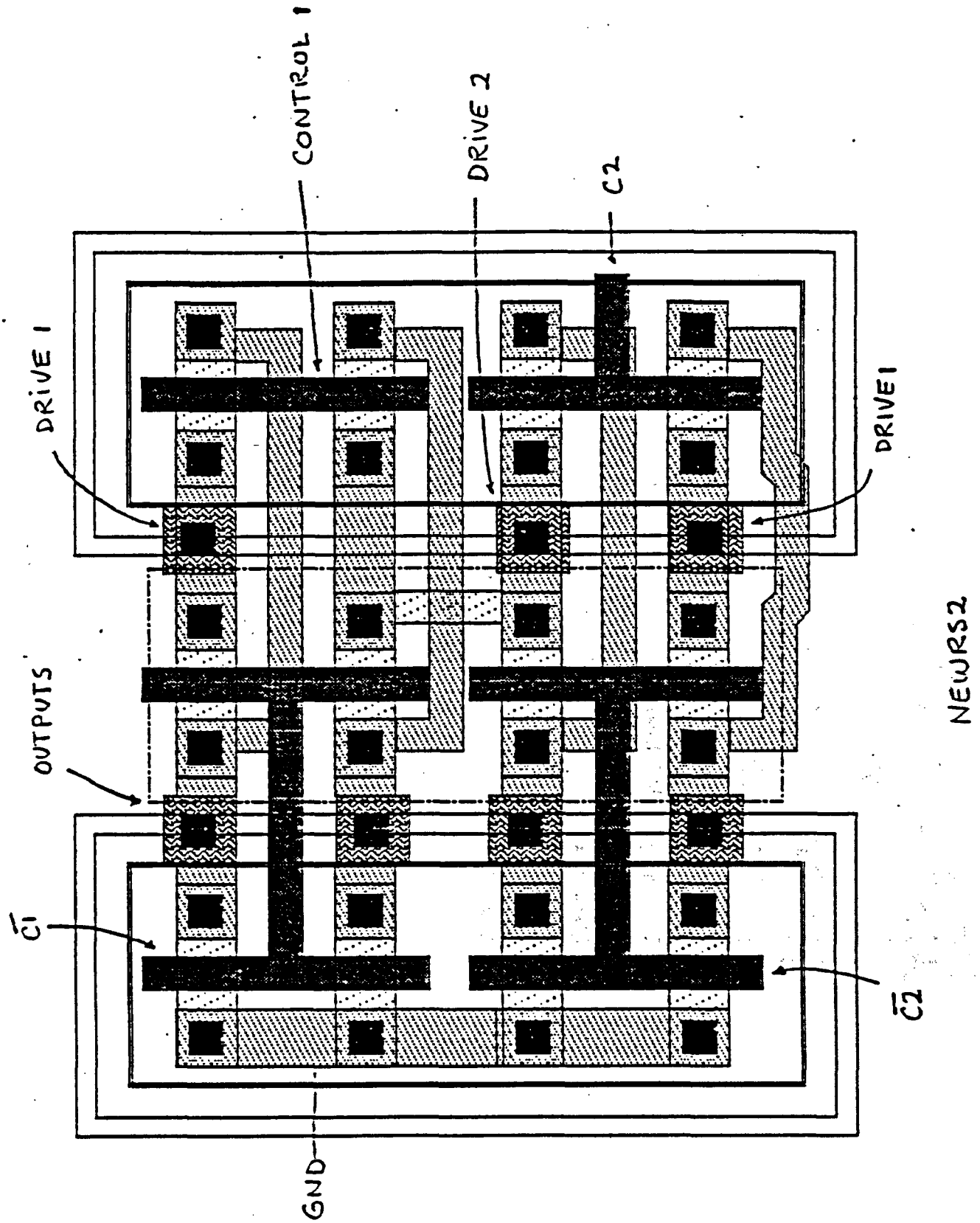
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



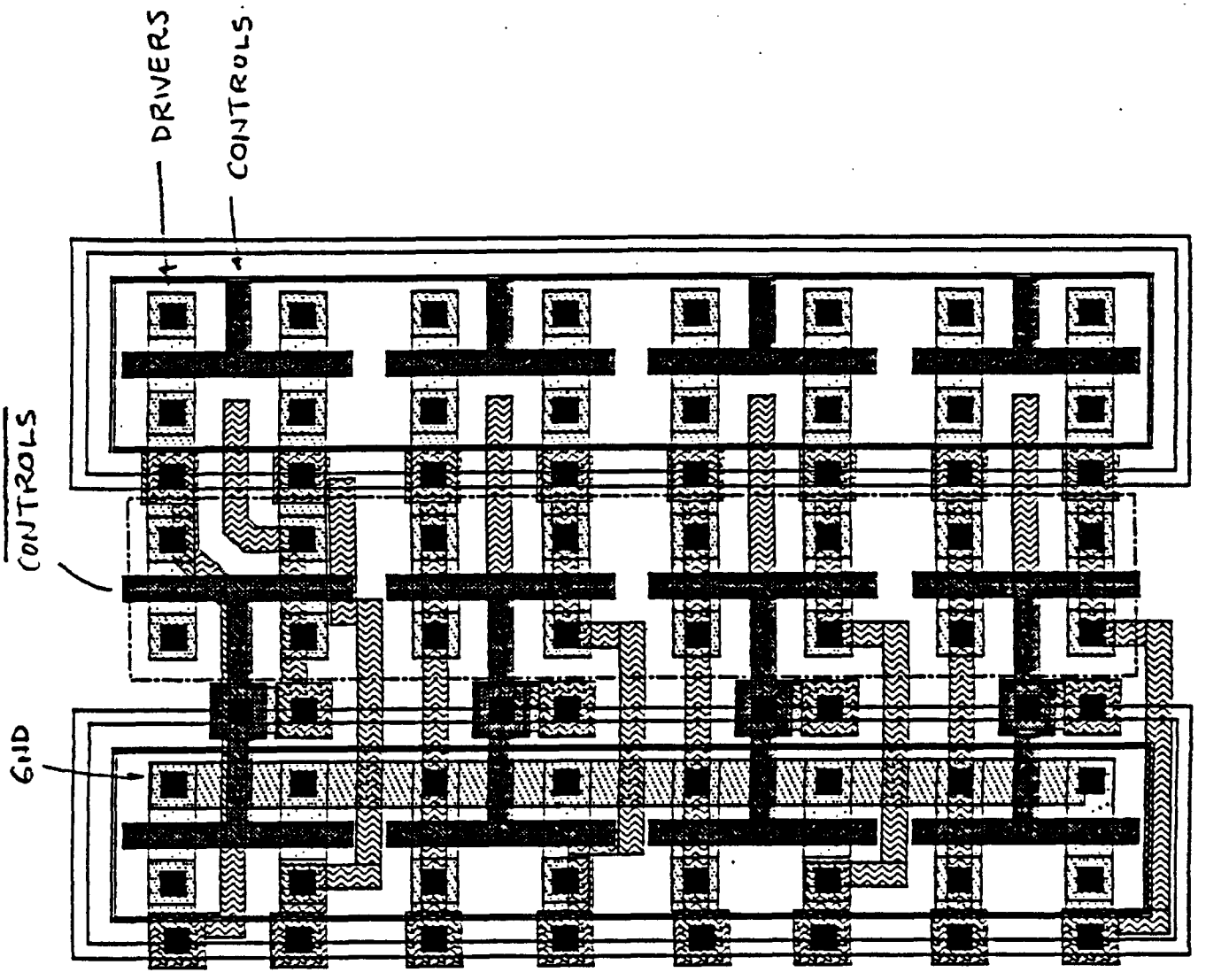
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



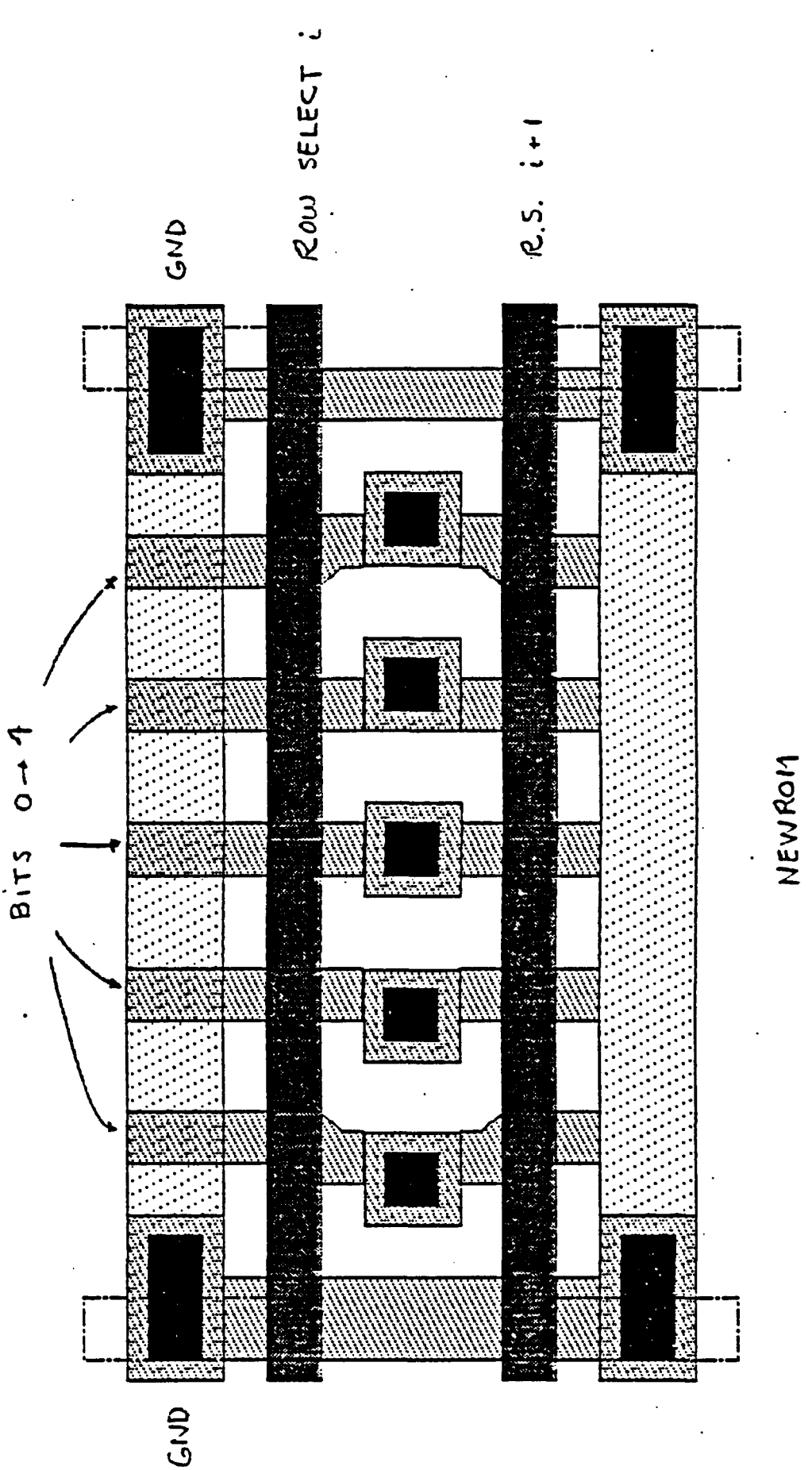
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



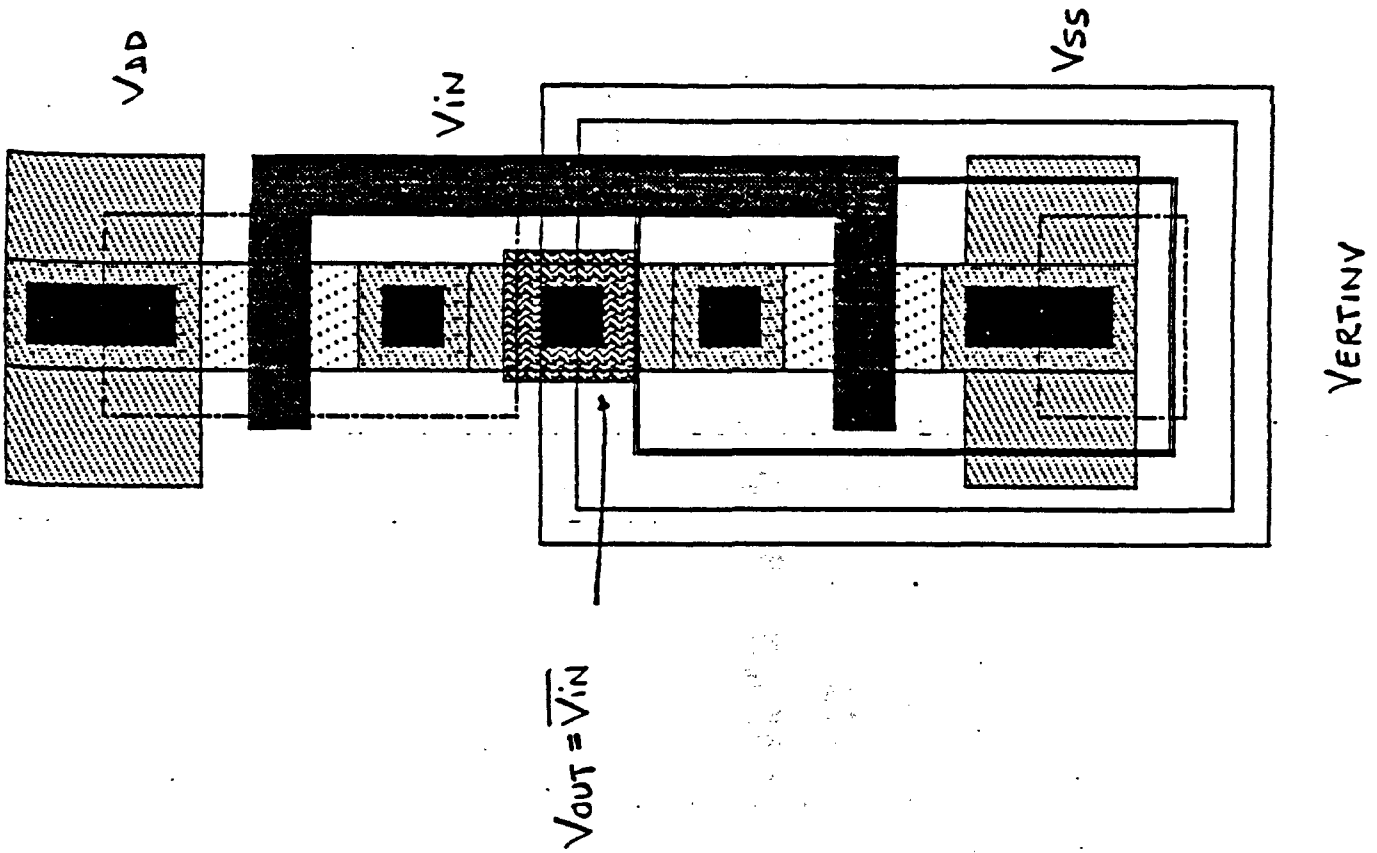
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



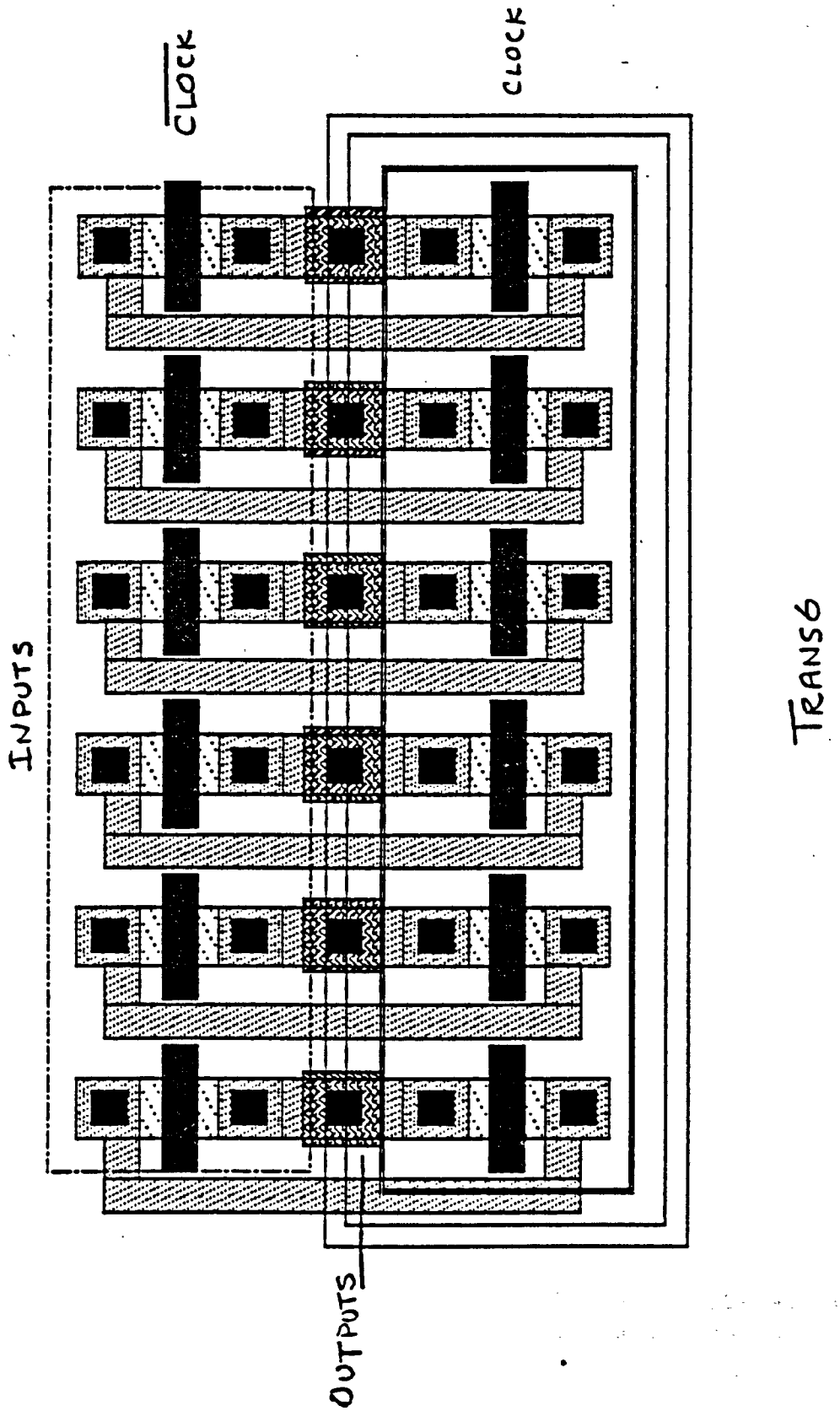
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

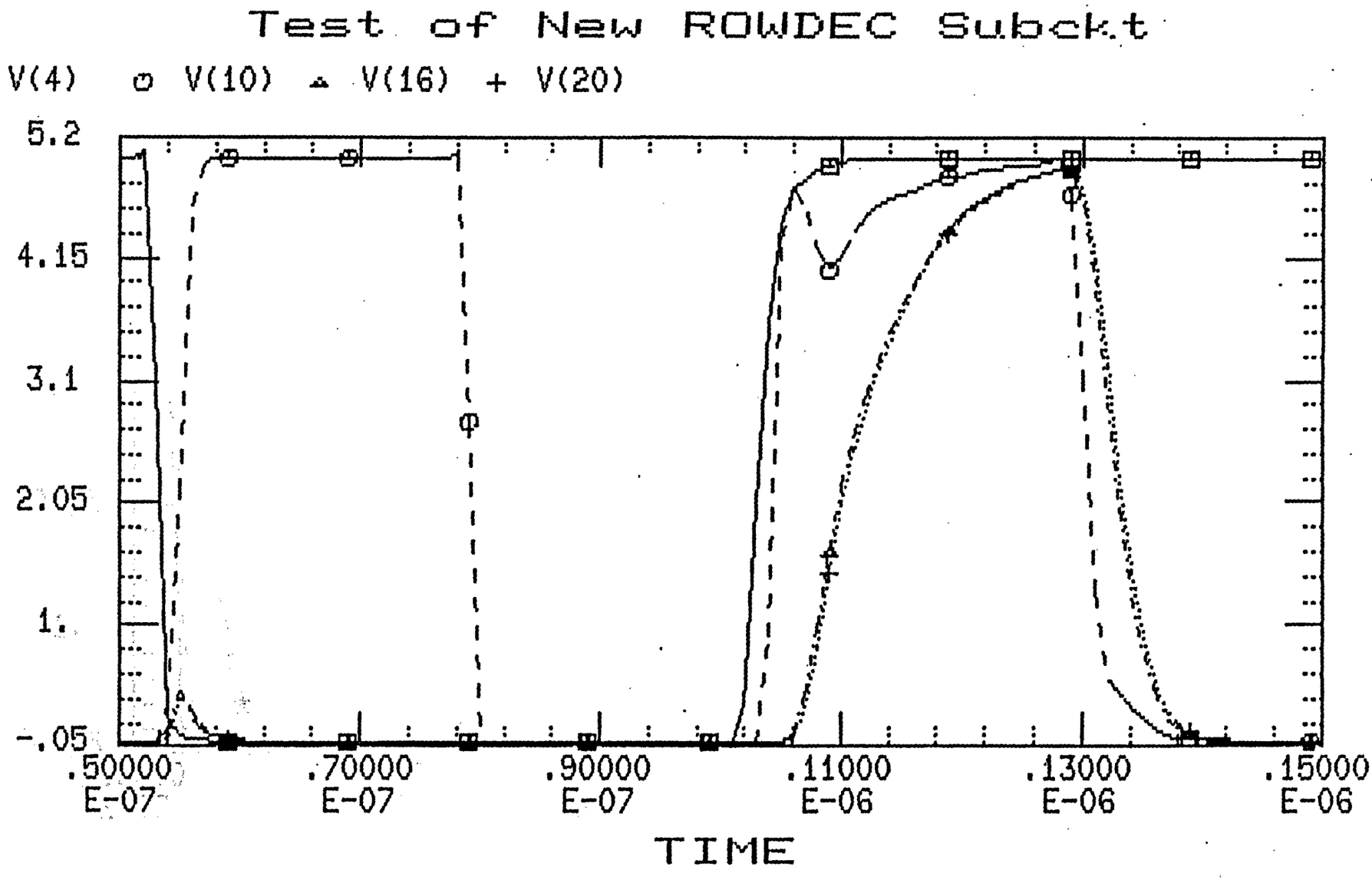
```
Test of New ROWDEC Subckt
***********************
* Start of Subcircuits
***********************
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*---------------
* Column Decoder
*---------------
* 1=VCC 2=VIN1 3=VIN2 4=Nand 5=And
.SUBCKT COLDEC 1 2 3 5 6
M1 5 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M2 5 3 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
M3 6 5 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
M4 5 3 4 0 NMOSE W=5.4U L=3.0U AD=39P AS=16P PD=25U PS=17U NRD=1.3 NRS=0.6
M5 4 2 0 0 NMOSE W=5.4U L=3.0U AD=16P AS=42P PD=17U PS=26U NRD=0.6 NRS=1.4
M6 6 5 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=42P PD=25U PS=26U NRD=1.3 NRS=1.4
.ENDS COLDEC
*-------------------------------
* Transmission gate matrix cell
*-------------------------------
* 1=Vcc 2=Input 3=ctl 4=ctl" 5=Output
.SUBCKT TGMAT 1 2 3 4 5
M1 2 4 5 1 PMOSE W=5.4U L=3.0U AD=39P AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M2 2 3 5 0 NMOSE W=5.4U L=3.0U AD=39P AS=24P PD=25U PS=20U NRD=1.3 NRS=0.8
M3 5 4 0 0 NMOSE W=5.4U L=3.0U AD=24P AS=62P PD=20U PS=34U NRD=0.8 NRS=2.1
.ENDS TGMAT
*---------------------------------
* row 1 subcircuit for binrom test
*---------------------------------
* 1=input 2=output
.SUBCKT ROW1 1 2
R1 1 2 449
C1 2 0 66.6FF
.ENDS ROW1
*---------------
* Input Inverter
*---------------
* 1=VCC 2=INPUT 3=OUTPUT
.SUBCKT INPINV 1 2 3
M1 3 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
M2 3 2 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
.ENDS INPINV
*--------------------------
* Row Select Drive Inverter
*--------------------------
* 1=VCC 2=INPUT 3=OUTPUT
.SUBCKT DRINV 1 2 3
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

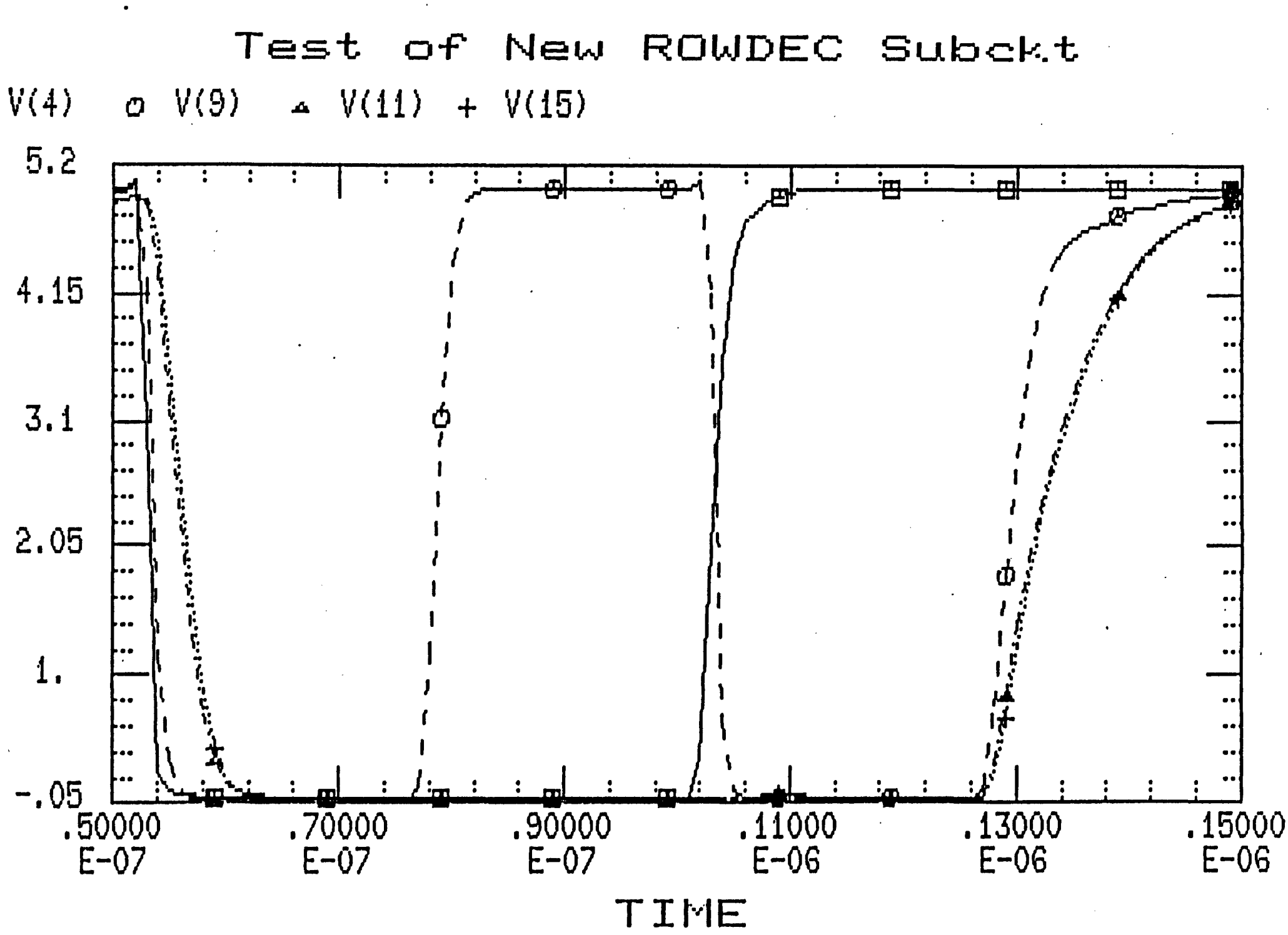
```
M1 3 2 1 1 PMOSE W=15.0U L=3.0U AD=65P AS=88P PD=46U PS=54U NRD=1.4 NRS=2.2
M2 3 2 0 0 NMOSE W=15.0U L=3.0U AD=65P AS=88P PD=46U PS=54U NRD=1.4 NRS=2.2
.ENDS DRINV
*********************
* Start of Circuits
*********************
VCC 1 0 DC 5

* 1=VCC 2=VIN1 3=VIN2 4=Nand 5=And
X1 1 4 4 6 5 COLDEC
* 1=Vcc 2=Input 3=ctl 4=ctl~ 5=Output
X2 1 9 5 6 11 TGMAT
X3 1 10 5 6 16 TGMAT
X4 1 0 2 3 RATINV
X5 1 0 7 8 RATINV
X6 1 3 4 INPINV
X7 1 8 9 DRINV
X8 1 9 10 DRINV
XR1 11 12 ROW1
XR2 12 13 ROW1
XR3 13 14 ROW1
XR4 14 15 ROW1
XR5 16 17 ROW1
XR6 17 18 ROW1
XR7 18 19 ROW1
XR8 19 20 ROW1
IL1 15 0 DC 0
IL2 20 0 DC 0

VR 7 0 PULSE(0 5 25NS 0 0 25NS 50NS)
VC 2 0 PULSE(0 5 0 0 0 50NS 100NS)

*****************************
* Start of Output Generation
*****************************
.TRAN 1.0NS 150NS 50NS
.PRINT TRAN V(4) V(9) V(11) V(15)
.PRINT TRAN V(1) V(10) V(16) V(20)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



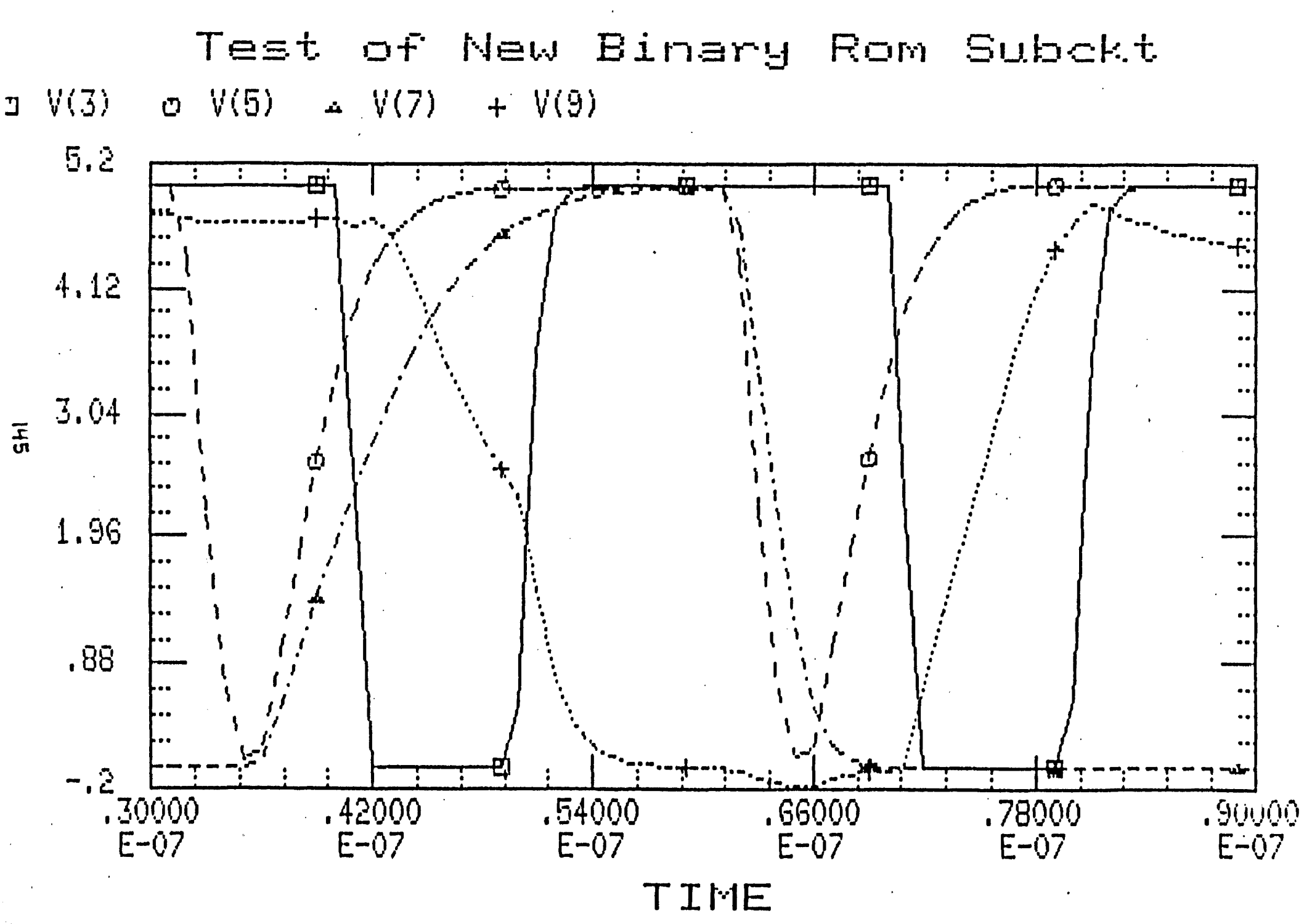
```
Test of New Binary Rom Subckt
*----------------------
* MOS TRANSISTOR MODELS
*----------------------
* DRAIN GATE SOURCE SUBSTRATE MOD1 L=3U W=3U
* nmos enhancement
.OPTIONS DEFW=3U DEFL=3U DEFAD=81P DEFAS=81P NOMOD
.MODEL NMOSE NMOS LEVEL=3 KP=50.0E-6 VTO=0.7 TOX=5E-8 GAMMA=1.1
+PHI=0.6 LAMBDA=0.01 RD=40 RS=40 PB=0.7 CGSO=3E-10 CGBO=5E-10
+CGDO=3E-10 RSH=25 CJ=44E-5 MJ=0.5 CJSW=4E-10 MJSW=0.3 JS=1E-5
+NSUB=1.7E16 XJ=6E-7 LD=3.5E-7 UO=775 VMAX=1E5 THETA=0.11
+ETA=0.05 KAPPA=1
* pmos enhancement
.MODEL PMOSE PMOS LEVEL=3 KP=16.0E-6 VTO=-0.8 TOX=5E-8 GAMMA=0.5
+PHI=0.6 LAMBDA=0.03 RD=100 RS=100 PB=0.6 CGSO=2.5E-10 CGBO=5E-10
+CGDO=2.5E-10 RSH=80 CJ=15E-5 MJ=0.6 CJSW=4E-10 MJSW=0.6 JS=1E-5
+NSUB=5E15 XJ=5E-7 LD=2.5E-7 UO=250 VMAX=0.7E5 THETA=0.12
+ETA=0.3 KAPPA=1

*************************
* Start of Subcircuits
*************************
*---------------
* Input Inverter
*---------------
* 1=VCC 2=INPUT 3=OUTPUT
.SUBCKT INPINV 1 2 3
M1 3 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
M2 3 2 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
.ENDS INPINV
*******************
* Start of Circuits
*******************
VCC 1 0 DC 5

X1 1 2 3 INPINV
X2 1 4 5 INPINV
X3 1 6 7 INPINV
M1 9 3 1 1 PMOSE W=7.2U L=3.0U AD=52P AS=48P PD=33U PS=28U NRD=1.0 NRS=1.0
M2 9 5 8 0 NMOSE W=5.4U L=3.0U AD=39P AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M3 8 7 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
CBIT 8 0 175FF
CL 9 0 100FF
CRS 7 0 260FF
CCS 5 0 124FF

VPR 2 0 PULSE(0 5 10NS 0 0 8NS 30NS)
VCOLS 4 0 PULSE(5 0 5NS 0 0 25NS 60NS)
VROWA 6 0 PULSE(5 0 35NS 0 0 25NS 60NS)

*****************************
* Start of Output Generation
*****************************
.TRAN 1.0NS 90NS 20NS
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
.PRINT ERAN [illegible]
.EMB
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



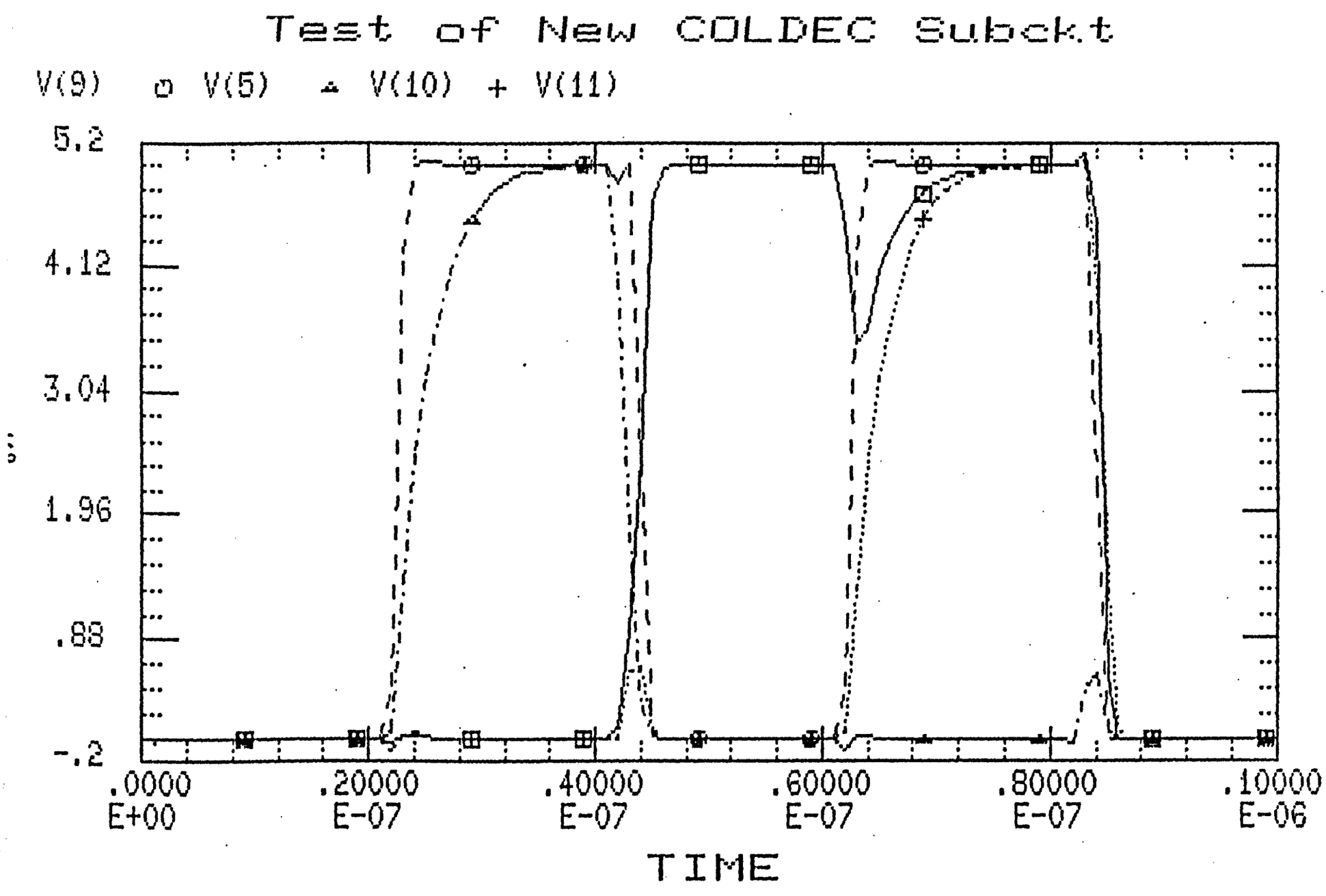
```
Test of New COLDEC Subckt
*-----------------------
* MOS TRANSISTOR MODELS
*-----------------------
* DRAIN GATE SOURCE SUBSTRATE MOD1 L=3U W=3U
* nmos enhancement
.OPTIONS DIEW=3U DIEL=3U DEFAD=81P DEFAS=81P NOMOD
.TEMP 40
.MODEL NMOSE NMOS LEVEL=2 KP=50.0E-6 VTO=0.7 TOX=5E-8 GAMMA=1.1
+PHI=0.6 LAMBDA=0.01 RD=40 RS=40 PB=0.7 CGSO=3E-10 CGBO=5E-10
+CGDO=3E-10 RSH=25 CJ=44E-5 MJ=0.5 CJSW=4E-10 MJSW=0.2 JS=1E-5
+NSUB=1.7E16 XJ=6E-7 LD=3.5E-7 UO=775 VMAX=1E5 THETA=0.11
+ETA=0.05 KAPPA=1
* pmos enhancement
.MODEL PMOSE PMOS LEVEL=2 KP=16.0E-6 VTO=-0.8 TOX=5E-8 GAMMA=0.6
+PHI=0.6 LAMBDA=0.03 RD=100 RS=100 PB=0.6 CGSO=2.5E-10 CGBO=5E-10
+CGDO=2.5E-10 RSH=80 CJ=15E-5 MJ=0.6 CJSW=4E-10 MJSW=0.6 JS=1E-5
+NSUB=5E15 XJ=5E-7 LD=2.5E-7 UO=250 VMAX=0.7E5 THETA=0.13
+ETA=0.3 KAPPA=1

*^^^^^^^^^^^^^^^^^^^^^^^^
* Start of Subcircuits
*^^^^^^^^^^^^^^^^^^^^^^^^
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*---------------
* Input Inverter
*---------------
* 1=VCC 2=INPUT 3=OUTPUT
.SUBCKT INPINV 1 2 3
M1 3 2 1 1 PMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
M2 3 2 0 0 NMOSE W=5.4U L=3.0U AD=39P AS=62P PD=25U PS=34U NRD=1.3 NRS=2.1
.ENDS INPINV
*-------------------------------
* Transmission gate matrix cell
*-------------------------------
* 1=Vcc 2=Input 3=ctl 4=ctl" 5=Output
.SUBCKT TGMAT 1 2 3 4 5
M1 2 4 5 1 PMOSE W=5.4U L=3.0U AD=39P AS=39P PD=25U PS=25U NRD=1.3 NRS=1.3
M2 2 3 5 0 NMOSE W=5.4U L=3.0U AD=39P AS=24P PD=25U PS=20U NRD=1.3 NRS=0.8
M3 5 4 0 0 NMOSE W=5.4U L=3.0U AD=24P AS=62P PD=20U PS=34U NRD=0.8 NRS=2.1
.ENDS TGMAT
*^^^^^^^^^^^^^^^^^^
* Start of Circuits
*^^^^^^^^^^^^^^^^^^
VCC 1 0 DC 5

X1 1 0 2 3 RATINV
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
X2 1 0 3 7 RATIO
X3 1 3 4 INPINV
X4 1 4 5 INPINV
X5 1 7 8 INPINV
X6 1 8 9 INPINV
* 1=Vcc 2=Input 3=ctl 4=ctl* 5=Output
X7 1 9 5 4 10 IGMAT
X8 1 9 5 4 11 IGMAT
C1 10 0 116FF
C2 11 0 116FF

V1 3 0 PULSE(5 0 20NS 0 0 20NS 40NS)
V2 4 0 PULSE(5 0 40NS 0 0 40NS 80NS)
*****************************
* Start of Output Generation
*****************************
.TRAN 1.0NS 100NS
.PRINT TRAN V(9) V(5) V(10) V(11)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Appendix IV

Ternary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Pin #	I/O	Function
1	I	A1
2	I	A0
3	I	Phi 1 bar
4	I	Phi 2
5	I	Phi 2 bar
6	I	Phi 1
7	I	Vdd
8	I	Vss
9	I	Gnd
10	O	B1
11	O	B0
12	O	B2
13	I	B2
14	I	B0
15	I	B1
16	I	A2

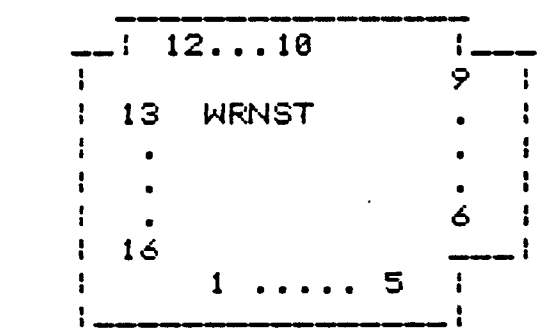
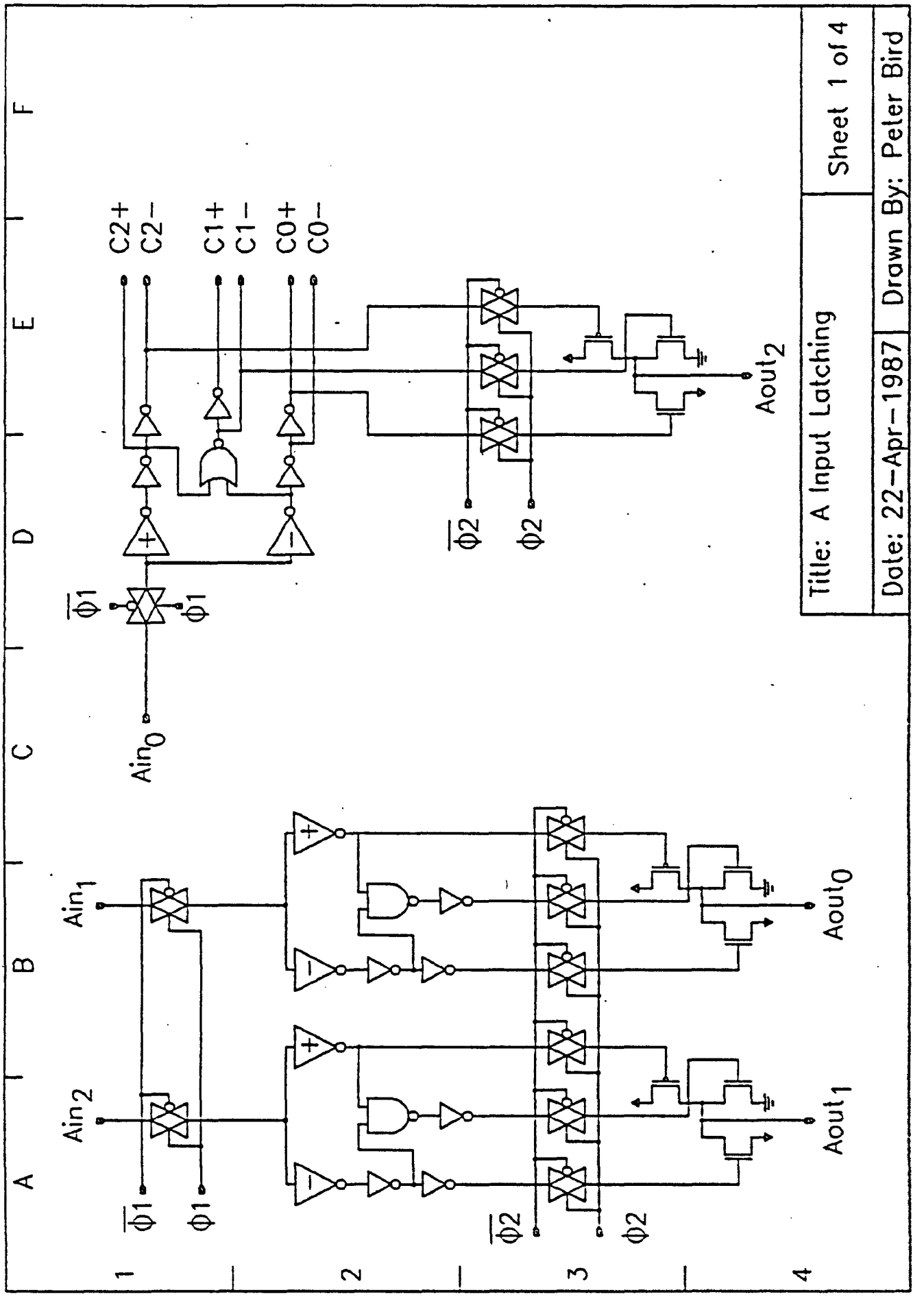


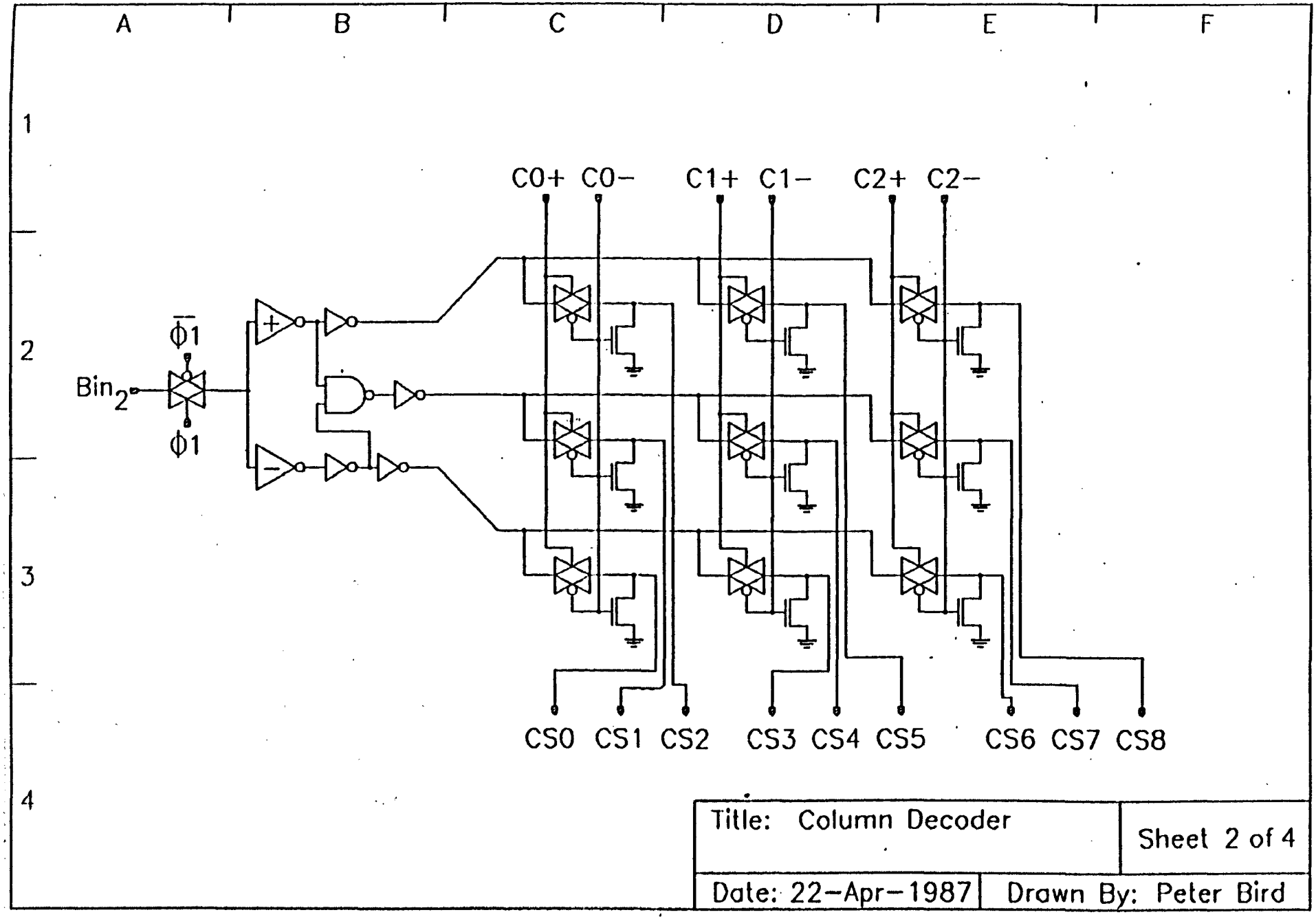
Figure A.4. Pinouts for the Improved Binary Design

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

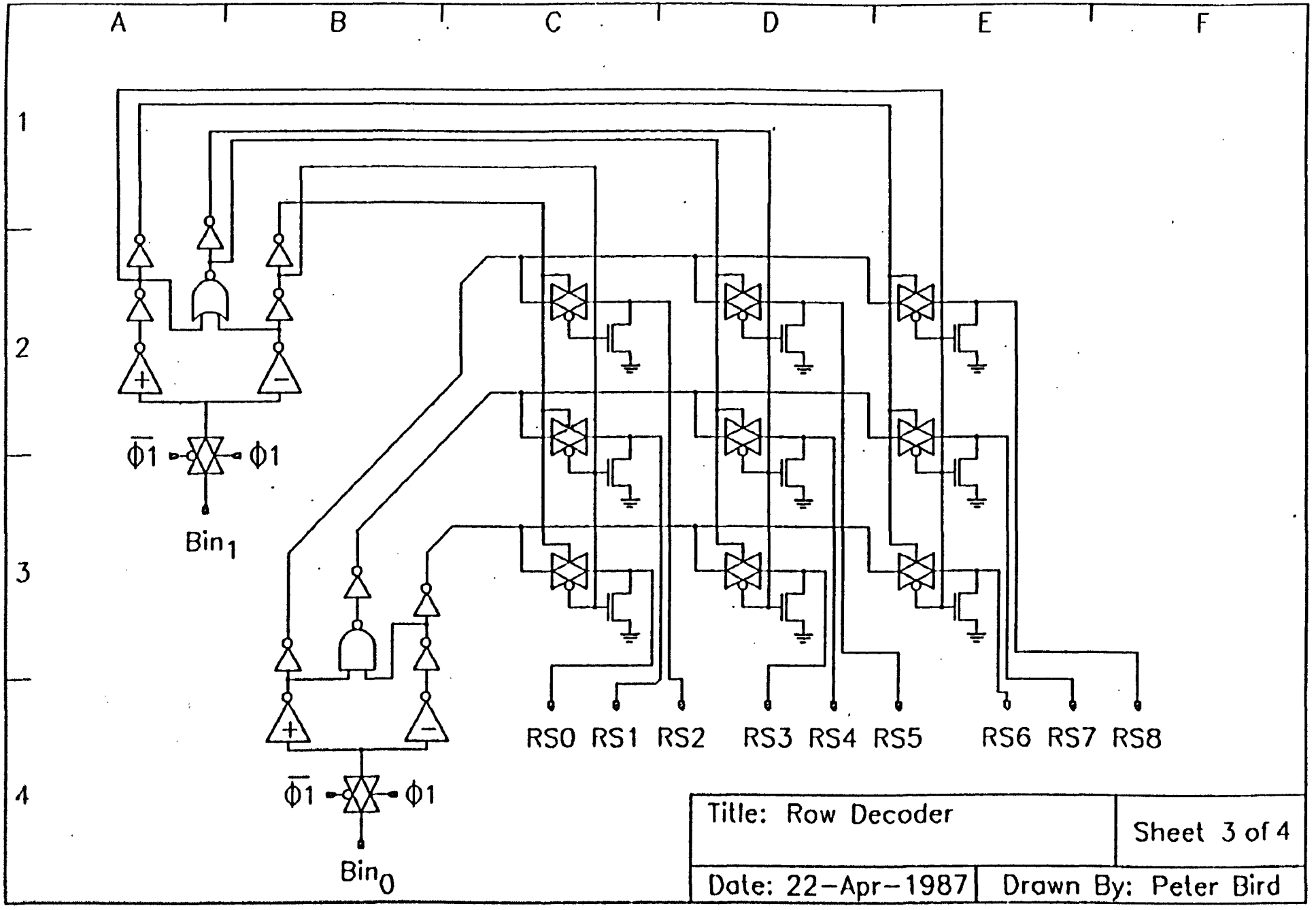


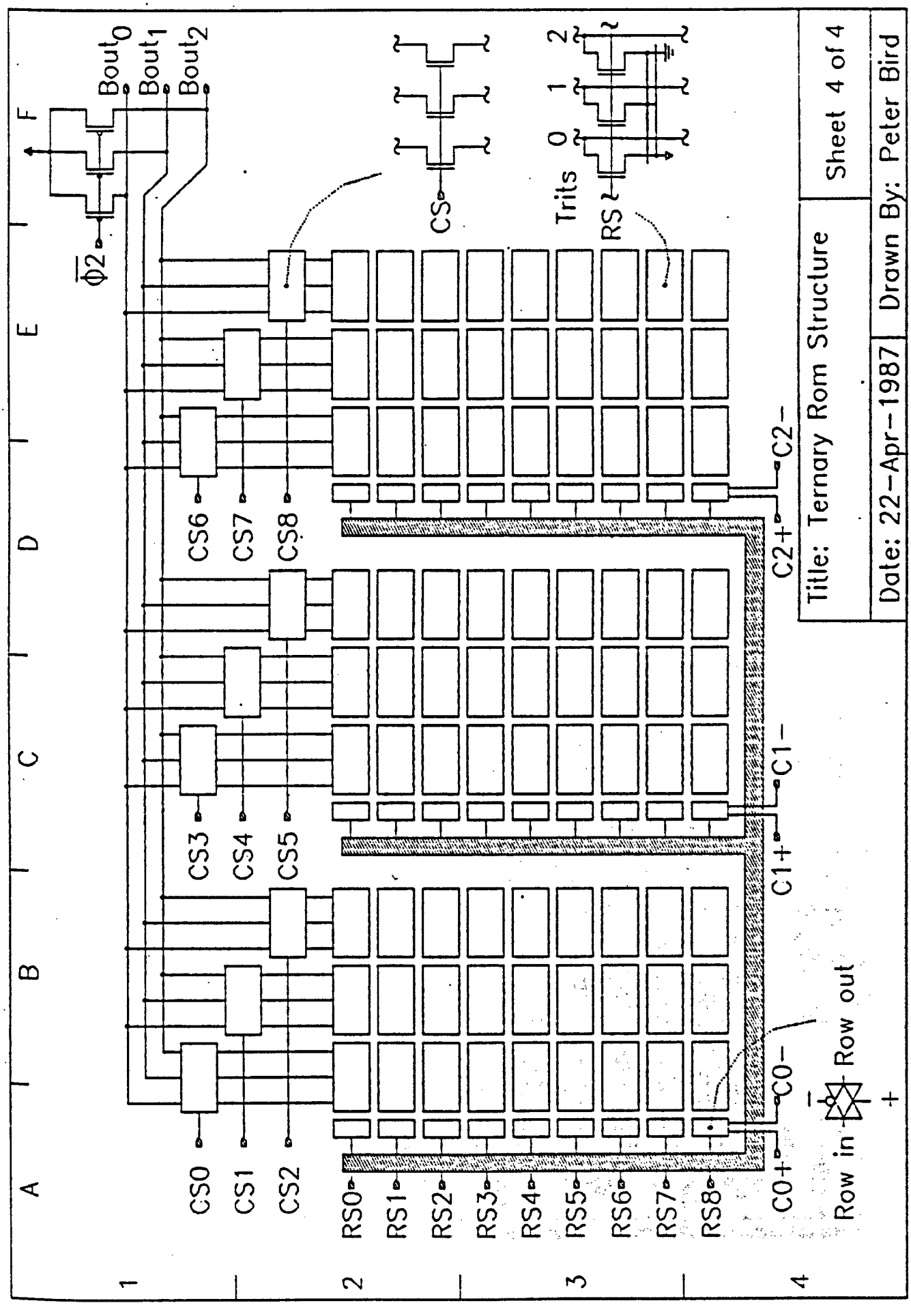
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

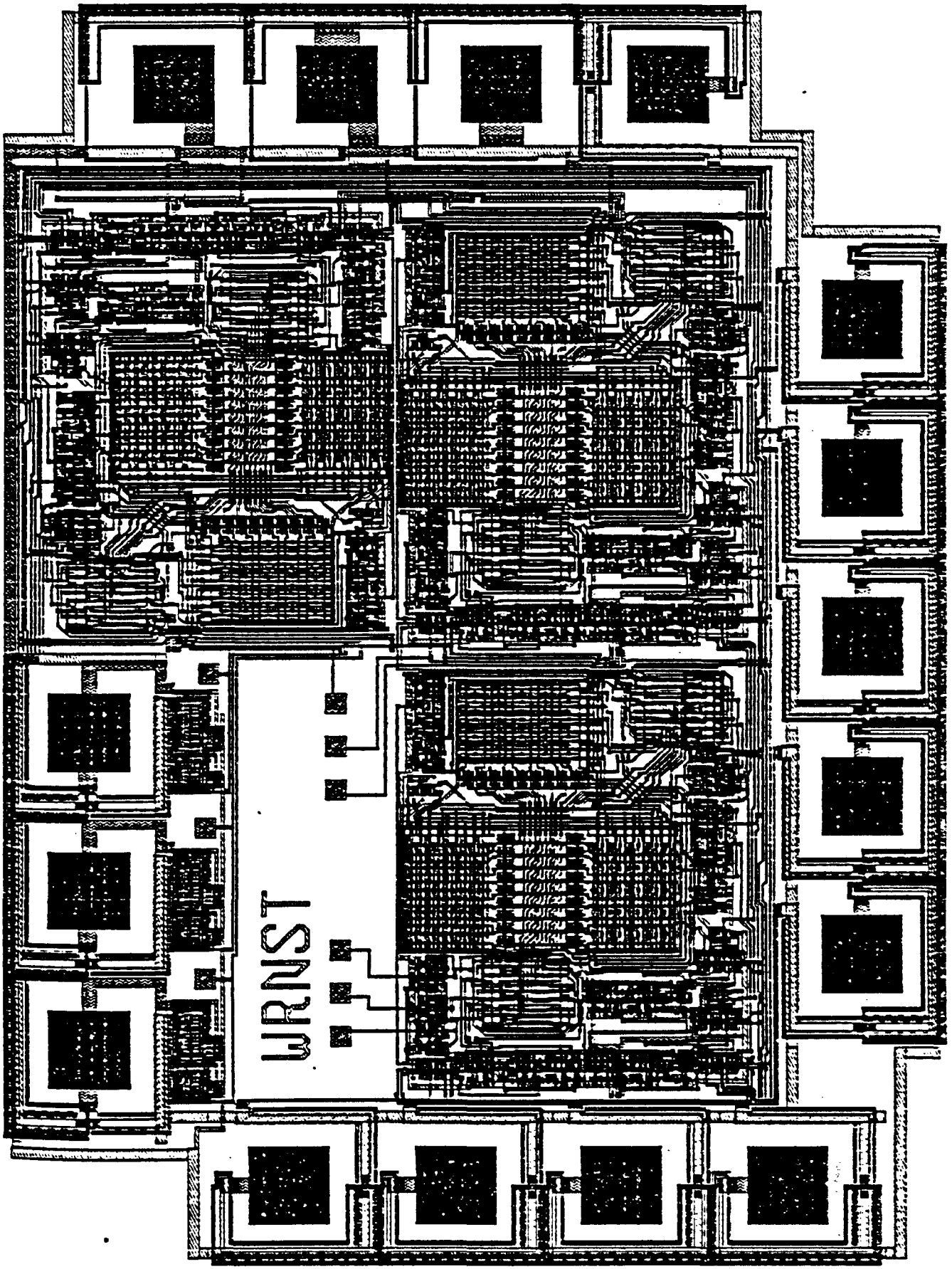


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

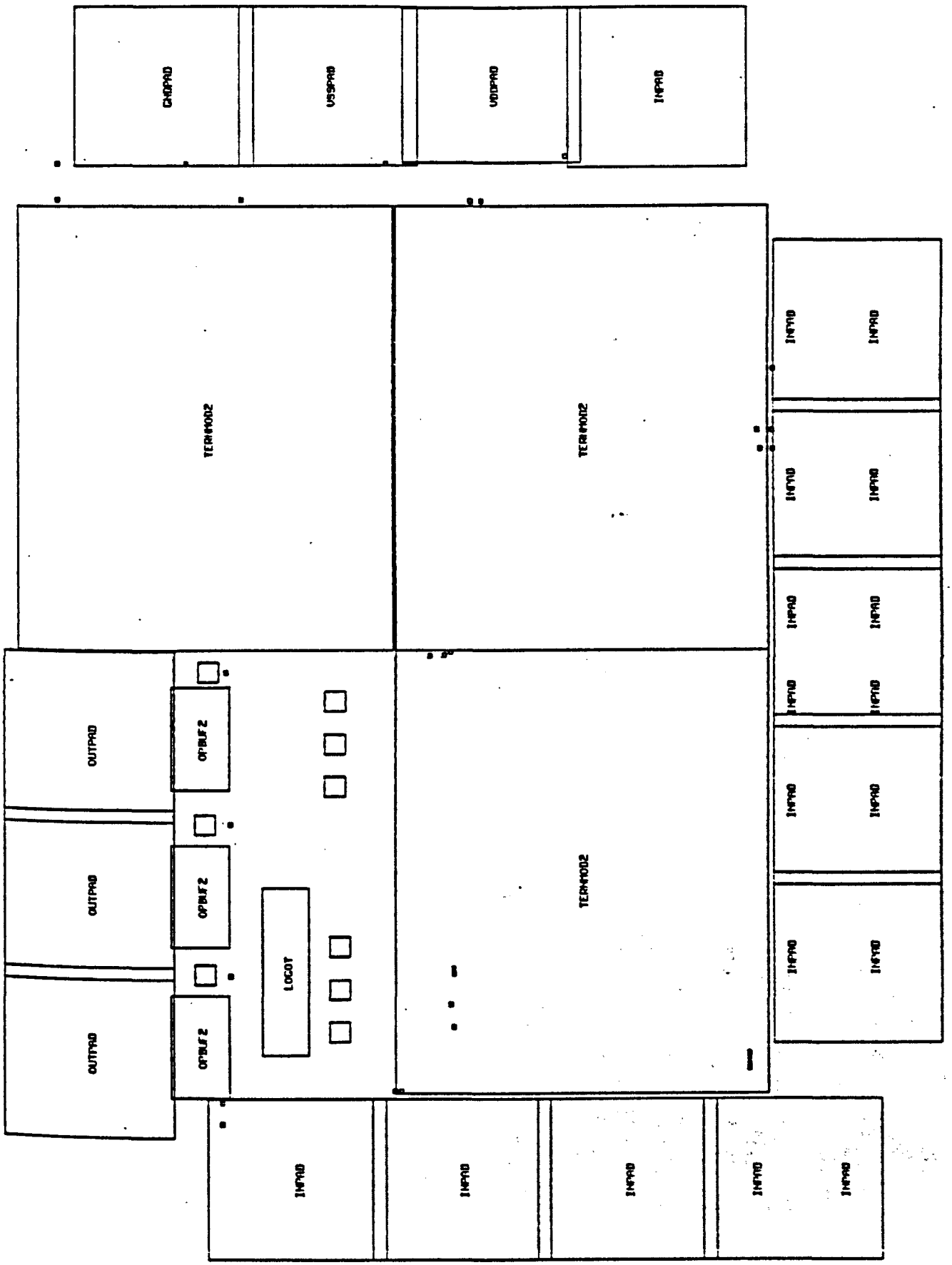




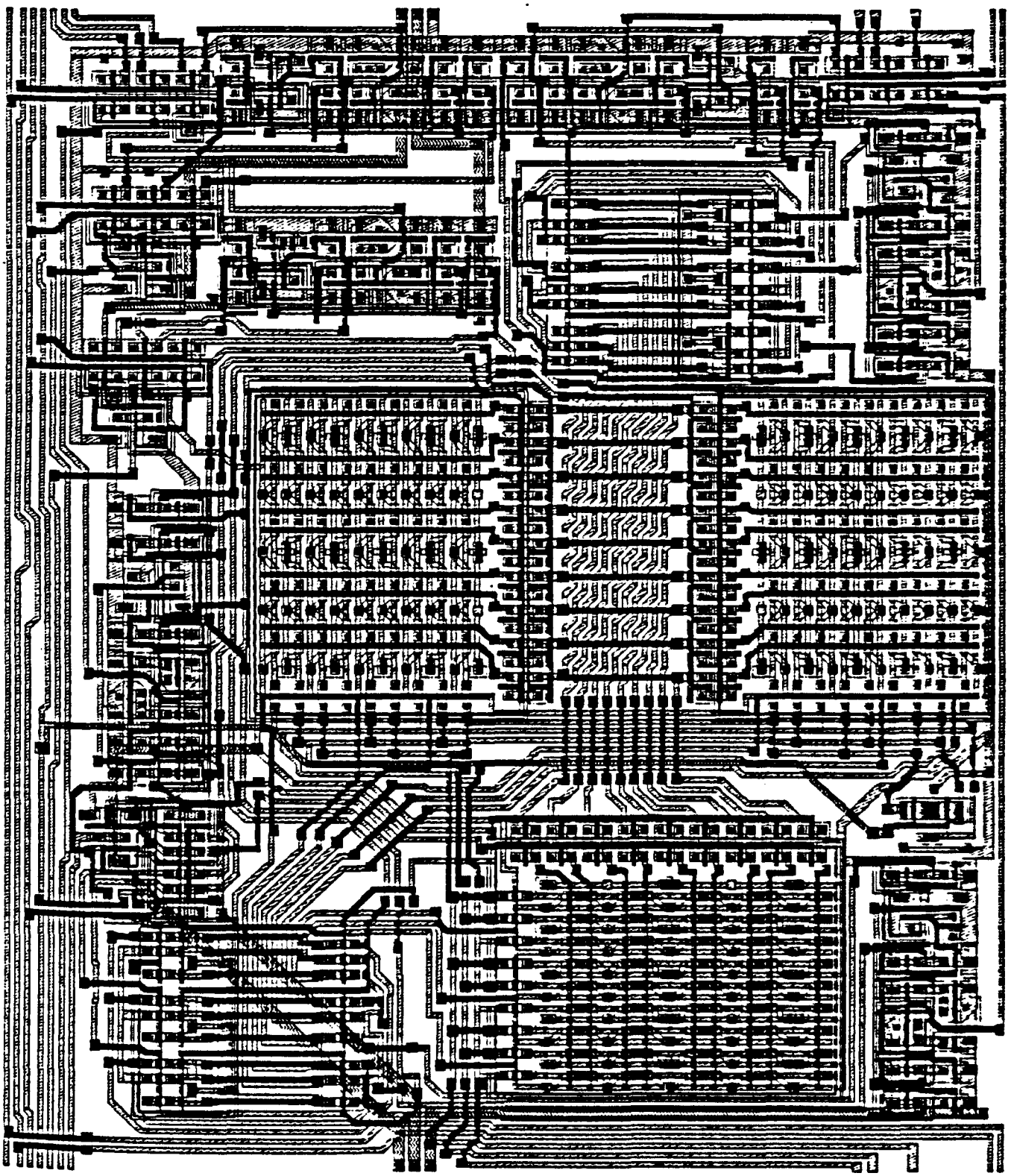
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



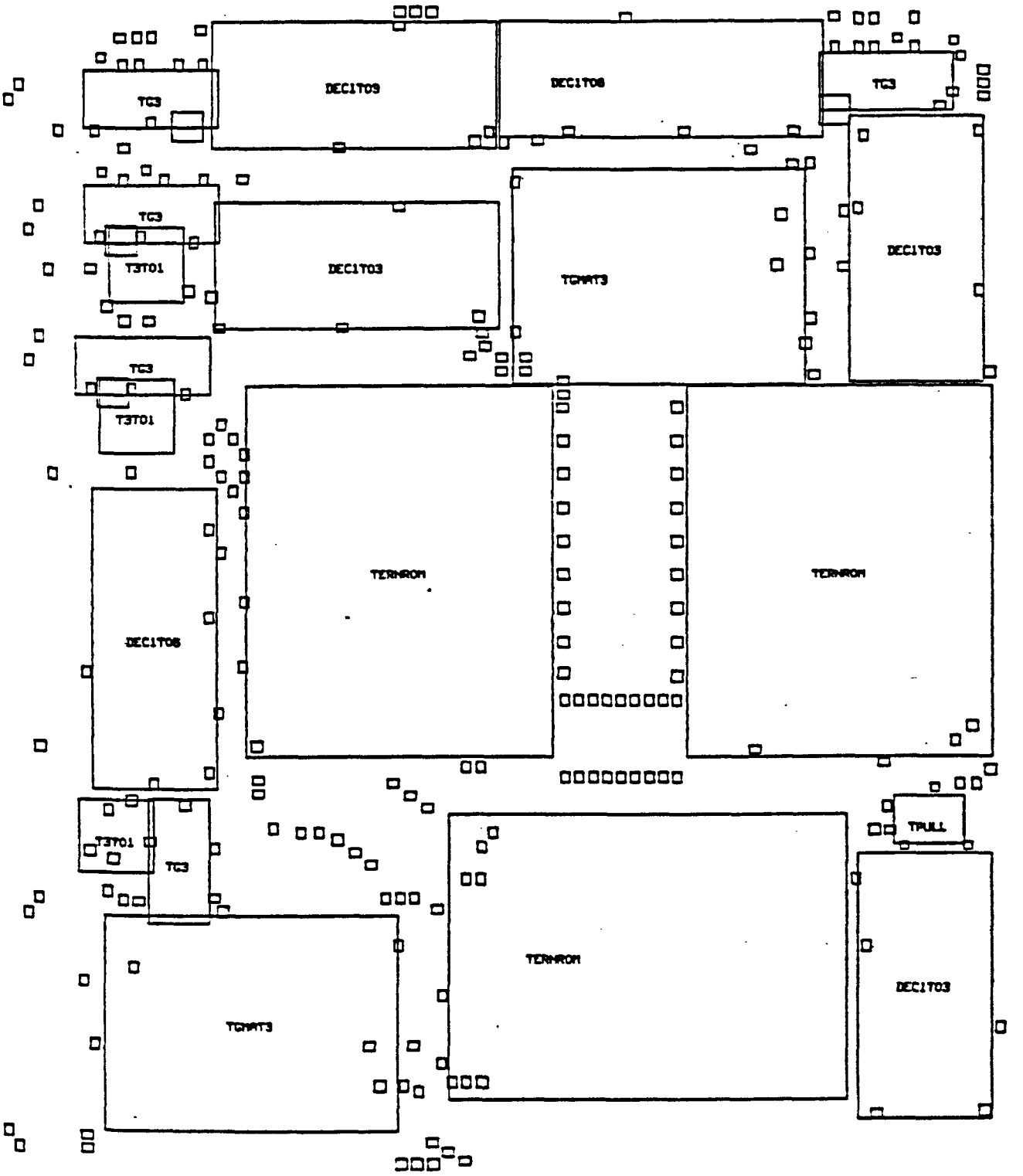
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



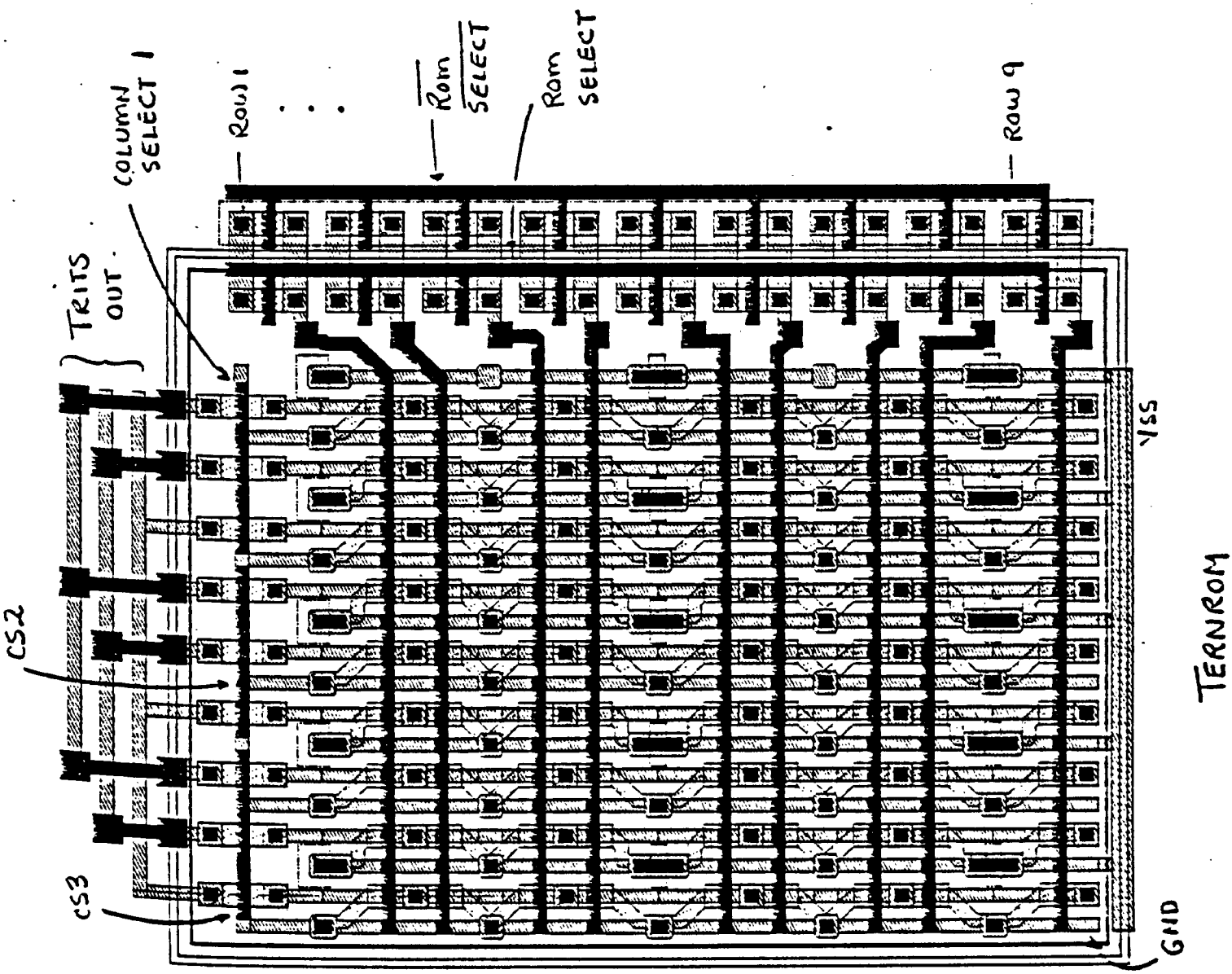
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



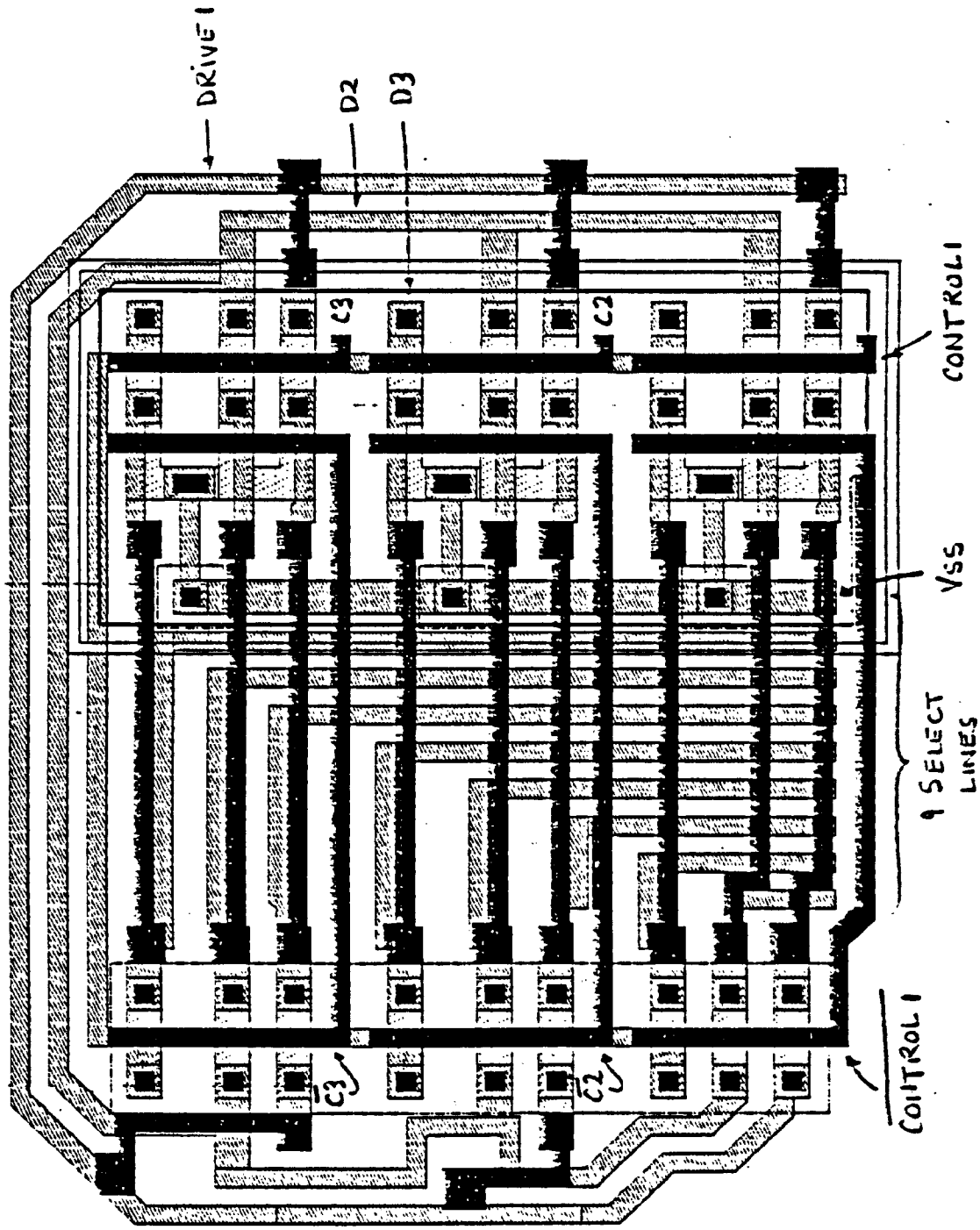
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

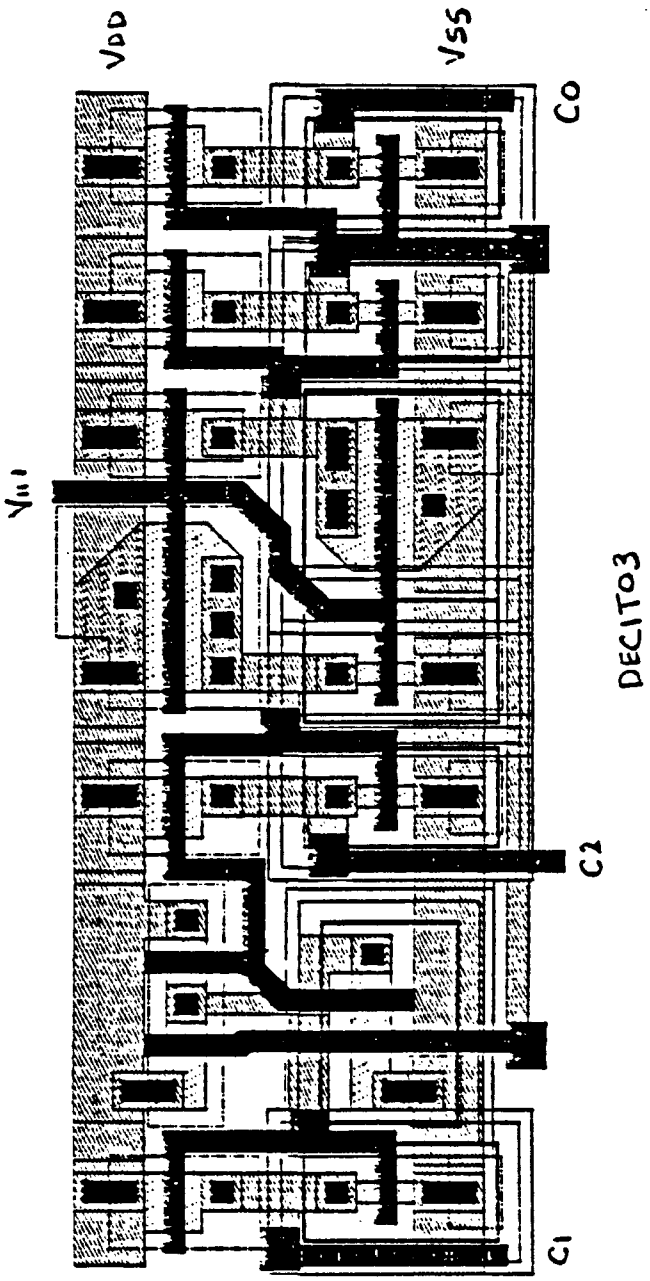


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

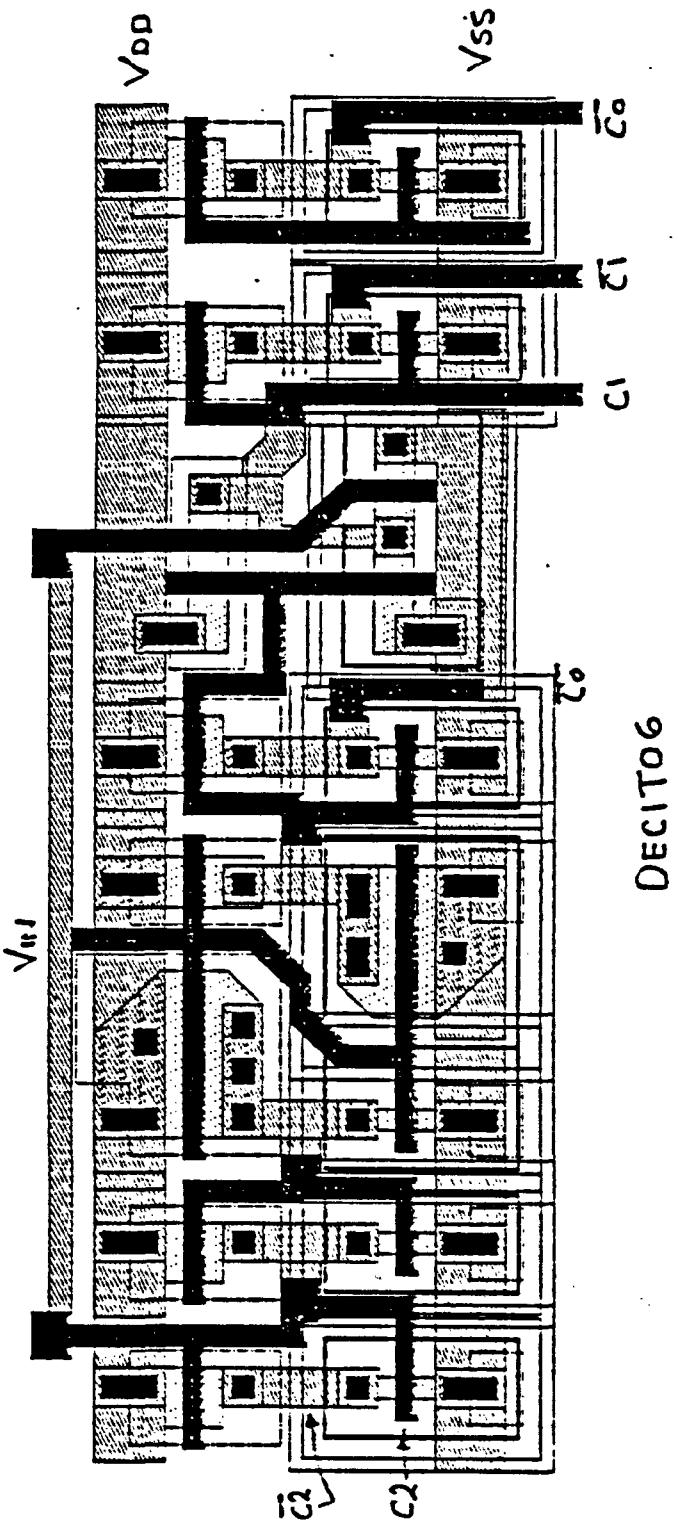


TGMAT

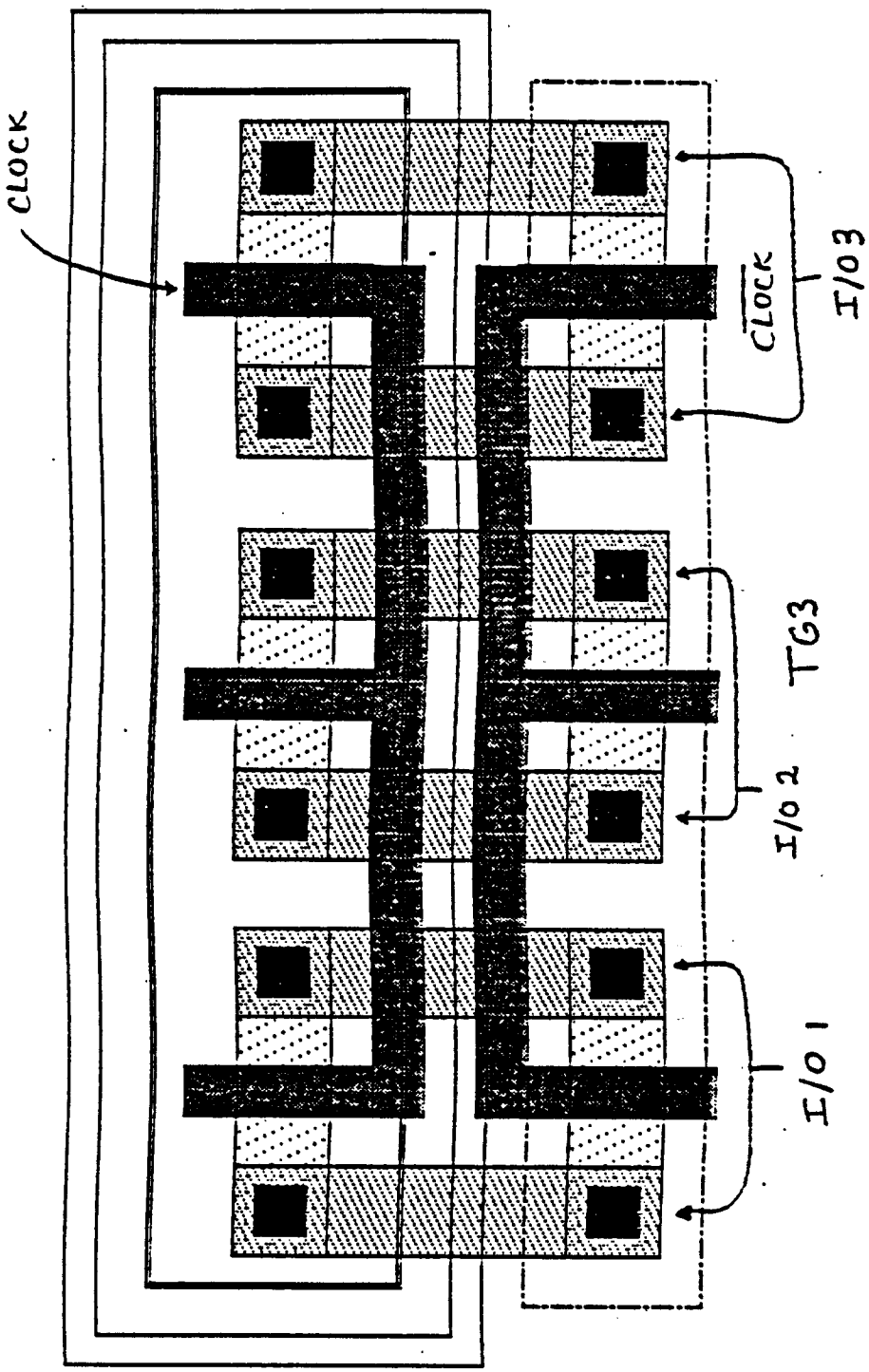
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



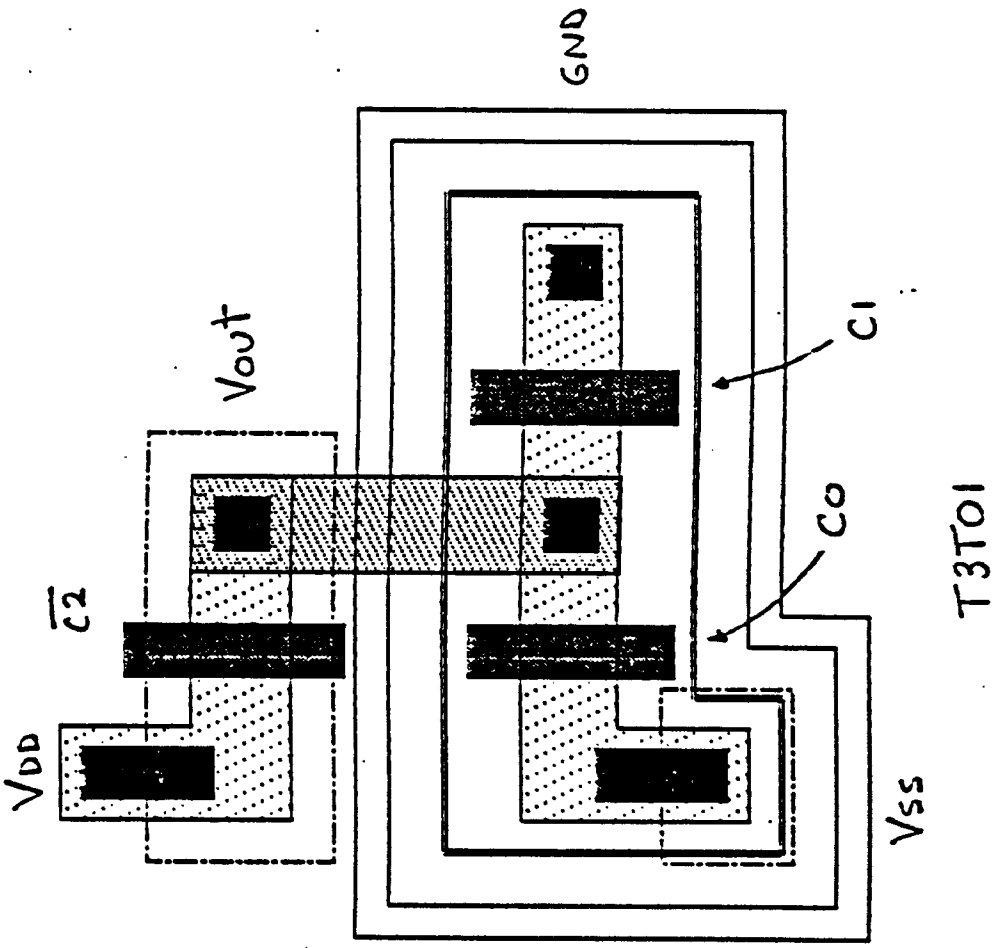
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

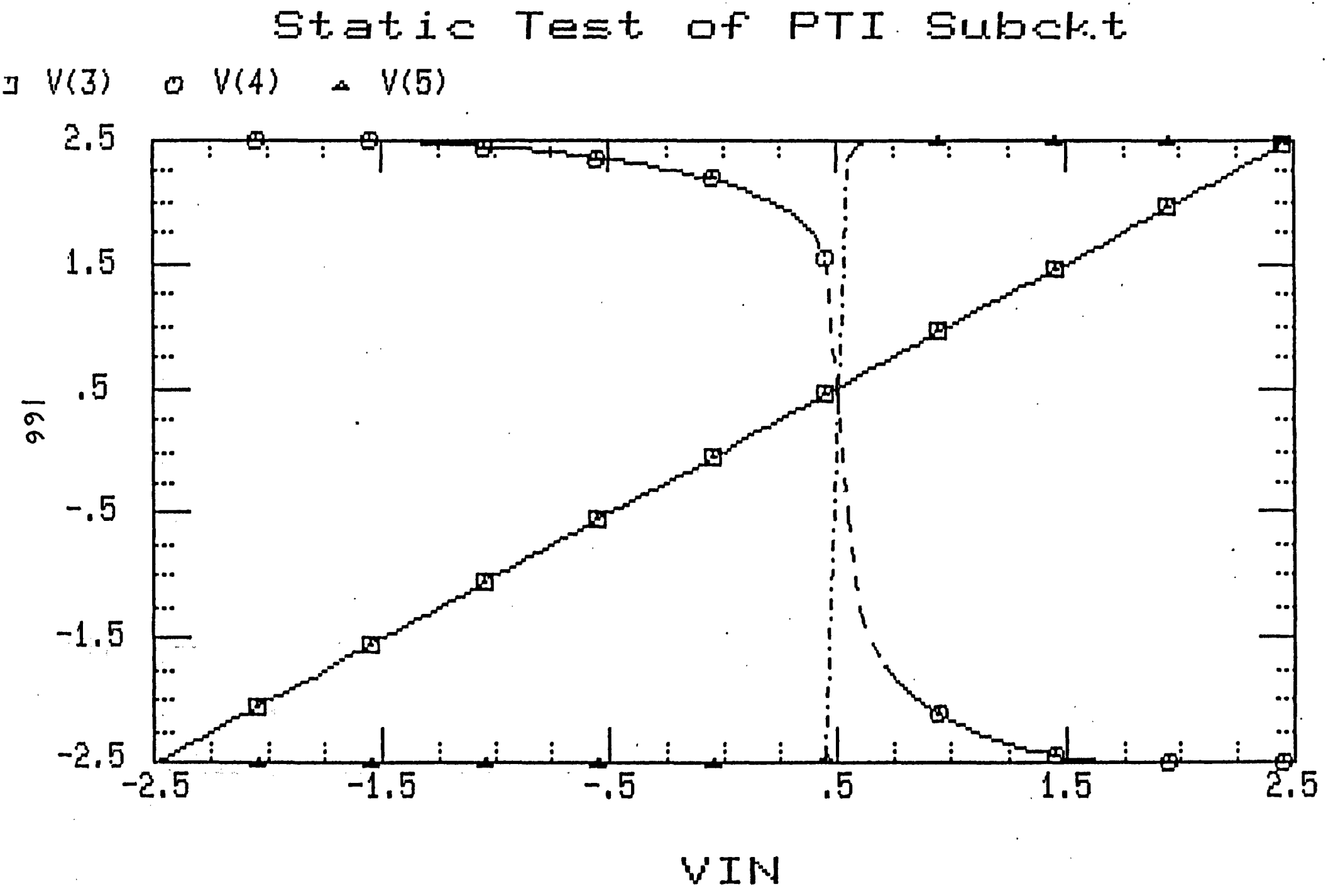


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

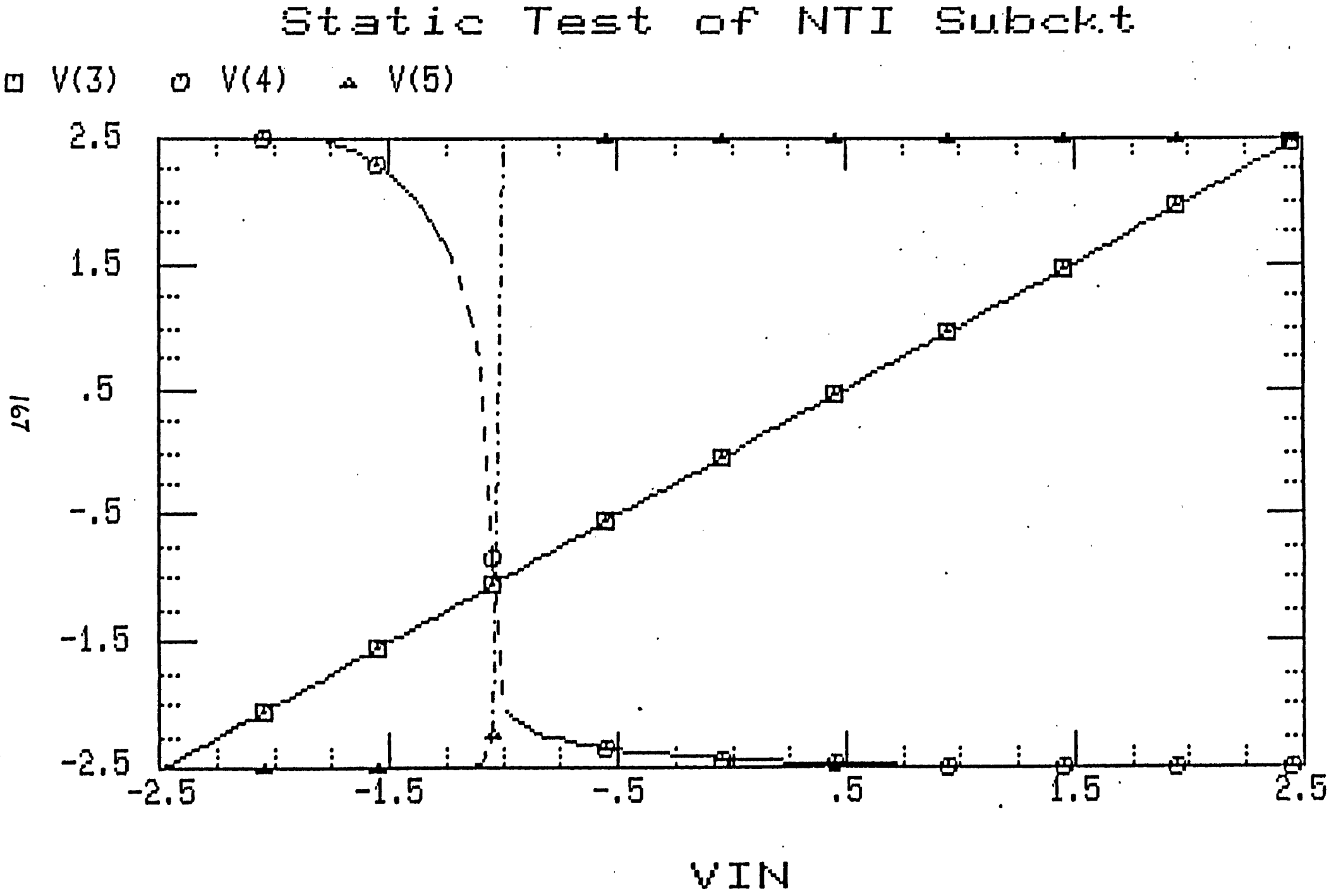
```
Static Test of NTI Subckt
*************************
* Start of Subcircuits
*************************
*---------------------------
* Negative ternary inverter
*---------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT NTI 1 2 3 4
M1 4 3 1 1 PMOSE W=7.2U  L=3.0U AD=50P  AS=75P  PD=30U PS=38U NRD=1.3 NRS=2.0
M2 4 3 2 2 NMOSE W=21.6U L=3.0U AD=149P AS=225P PD=59U PS=67U NRD=0.4 NRS=0.7
.ENDS NTI
*************************
* Start of Circuits
*************************
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5
X1 1 2 3 4 NTI
X2 1 2 4 5 NTI
VIN 3 0 DC 0
*****************************
* Start of Output Generation
*****************************
.DC VIN -2.5 2.5 0.05
.PRINT DC V(3) V(4) V(5)
.END


Static Test of PTI Subckt
*************************
* Start of Subcircuits
*************************
*---------------------------
* Positive ternary inverter
*---------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT PTI 1 2 3 4
M1 4 3 1 1 PMOSE W=24U L=3U PD=62U PS=62U AD=187P AS=56P NRD=0.5 NRS=0.3
M2 4 3 2 2 NMOSE W=3U  L=3U PD=26U PS=35U AD=37P  AS=60P NRD=2.6 NRS=4.0
.ENDS PTI
*************************
* Start of Circuits
*************************
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5
X1 1 2 3 4 PTI
X2 1 2 4 5 PTI
VIN 3 0 DC 0
*****************************
* Start of Output Generation
*****************************
.DC VIN -2.5 2.5 0.05
.PRINT DC V(3) V(4) V(5)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



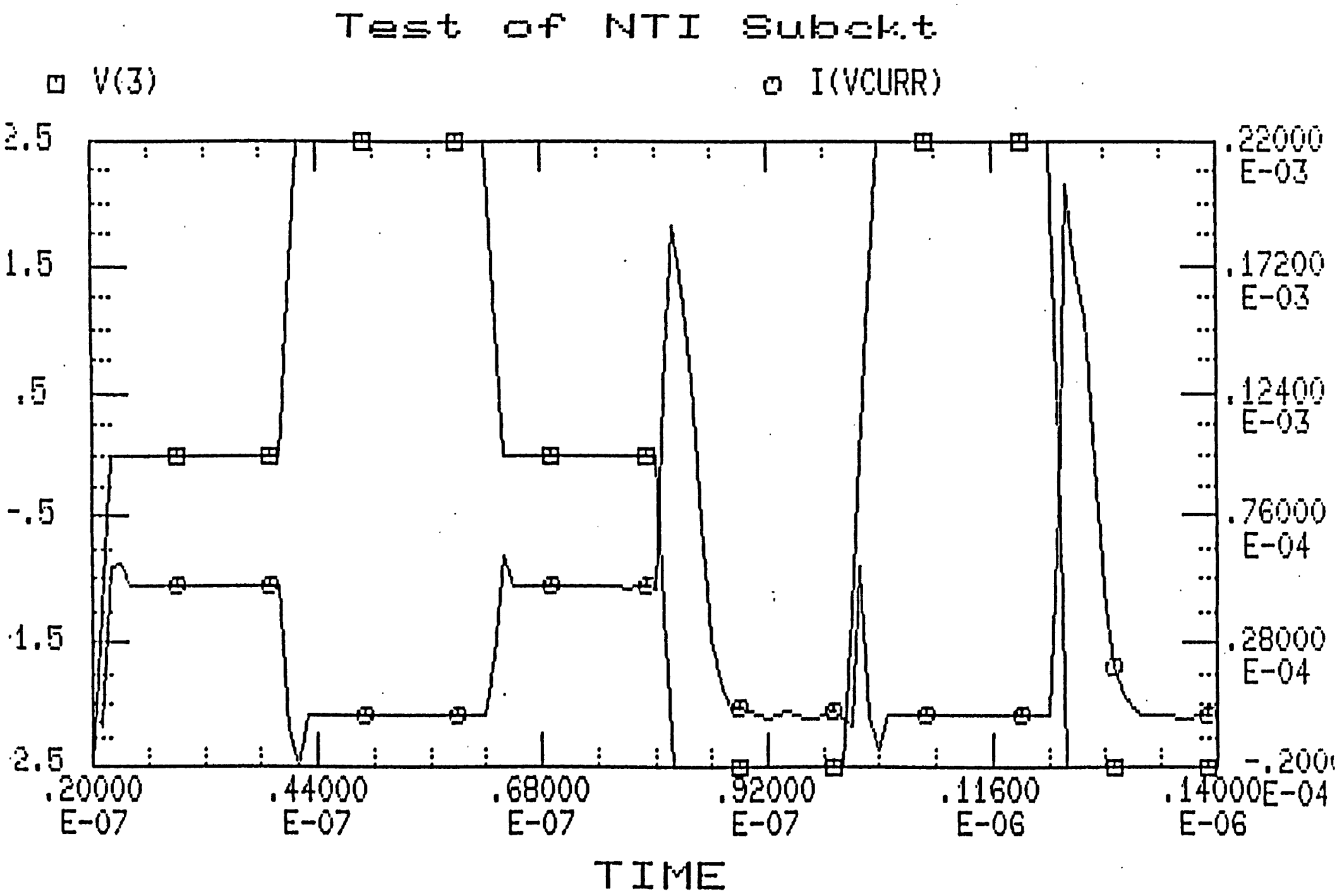
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of NTI Subckt
*************************
* Start of Subcircuits
*************************
*---------------------------
* Negative ternary inverter
*---------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT NTI 1 2 3 4
M1 4 3 1 1 PMOSE W=7.2U  L=3.0U AD=50P  AS=75P  PD=30U PS=38U NRD=1.3 NRS=1.9
M2 4 3 2 2 NMOSE W=21.6U L=3.0U AD=149P AS=225P PD=59U PS=67U NRD=0.4 NRS=0.7
.ENDS NTI
*---------------------------------------
* Test Power Supply for Ternary Circuits
*---------------------------------------
* 1=OUTPUT
.SUBCKT SOURCE 1
VIN1 1 2 PULSE( 0  2.5 40NS 2NS 2NS 20NS  60NS)
VIN2 2 3 PULSE(-2.5  0 20NS 2NS 2NS 100NS 120NS)
VIN3 3 0 PULSE( 0 -2.5 80NS 2NS 2NS 20NS  120NS)
.ENDS SOURCE
*********************
* Start of Circuits
*********************
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5

VCURR 1 6 DC 0
X1 6 2 3 4 NTI
X2 1 2 4 5 NTI
XSRC 3 SOURCE

*****************************
* Start of Output Generation
*****************************
.TRAN 1NS 140NS 20NS
.PRINT TRAN V(3) I(VCURR)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

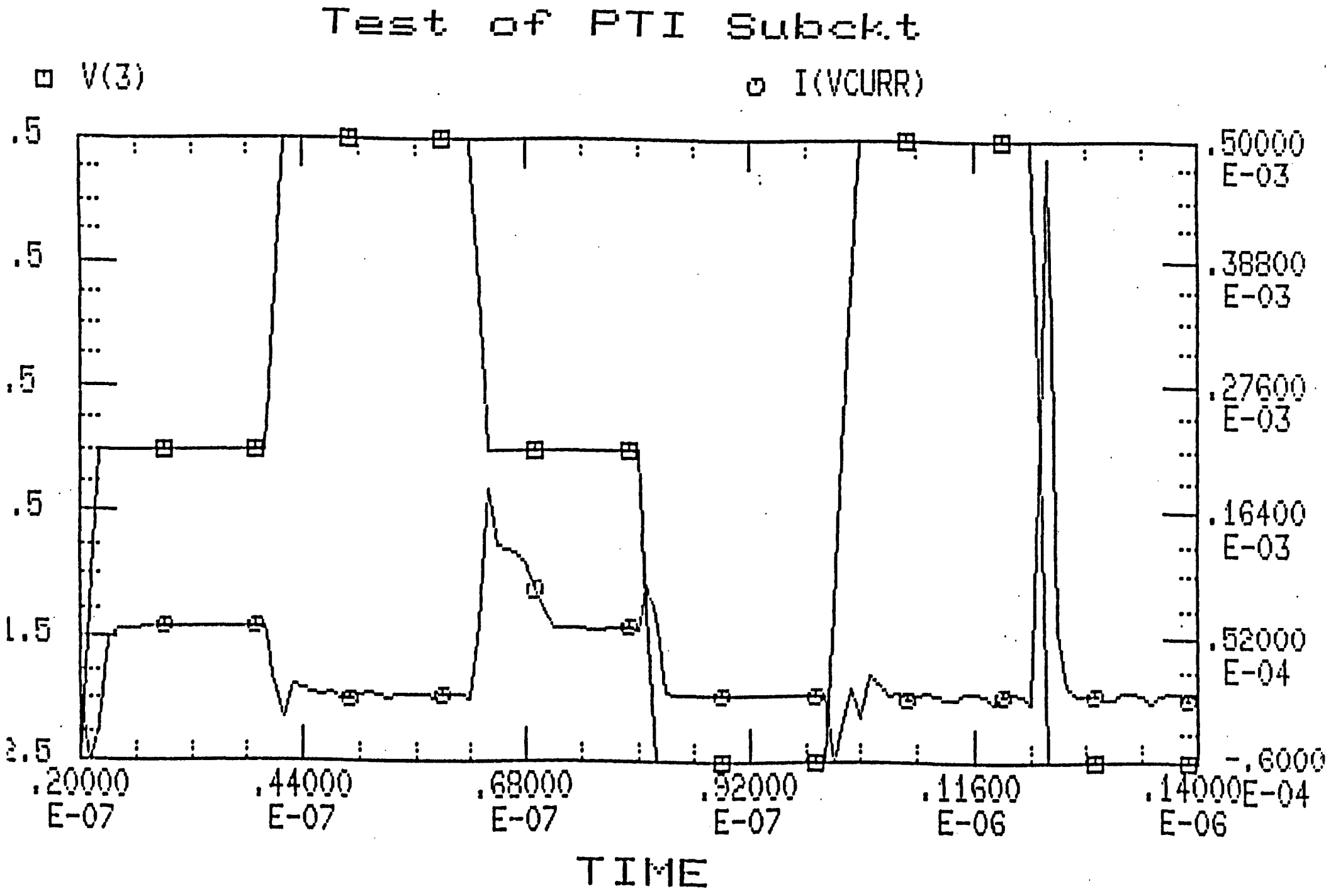
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of PTI Subckt
*************************
* Start of Subcircuits
*************************
*-------------------------
* Positive ternary inverter
*-------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT PTI 1 2 3 4
M1 4 3 1 1 PMOSE W=24U L=3U PD=62U PS=62U AD=187P AS=56P NRD=0.5 NRS=0.3
M2 4 3 2 2 NMOSE W=3U  L=3U PD=26U PS=35U AD=37P  AS=60P NRD=2.6 NRS=4.0
.ENDS PTI
*---------------------------------------
* Test Power Supply for Ternary Circuits
*---------------------------------------
* 1=OUTPUT
.SUBCKT SOURCE 1
VIN1 1 2 PULSE( 0  2.5 40NS 2NS 2NS 20NS  60NS)
VIN2 2 3 PULSE(-2.5  0 20NS 2NS 2NS 100NS 120NS)
VIN3 3 0 PULSE( 0 -2.5 80NS 2NS 2NS 20NS  120NS)
.ENDS SOURCE

*********************
* Start of Circuits
*********************
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5

VCURR 1 6 DC 0
X1 6 2 3 4 PTI
X2 1 2 4 5 PTI
XSRC 3 SOURCE

*****************************
* Start of Output Generation
*****************************
.TRAN 1NS 140NS 20NS
.PRINT TRAN V(3) I(VCURR)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

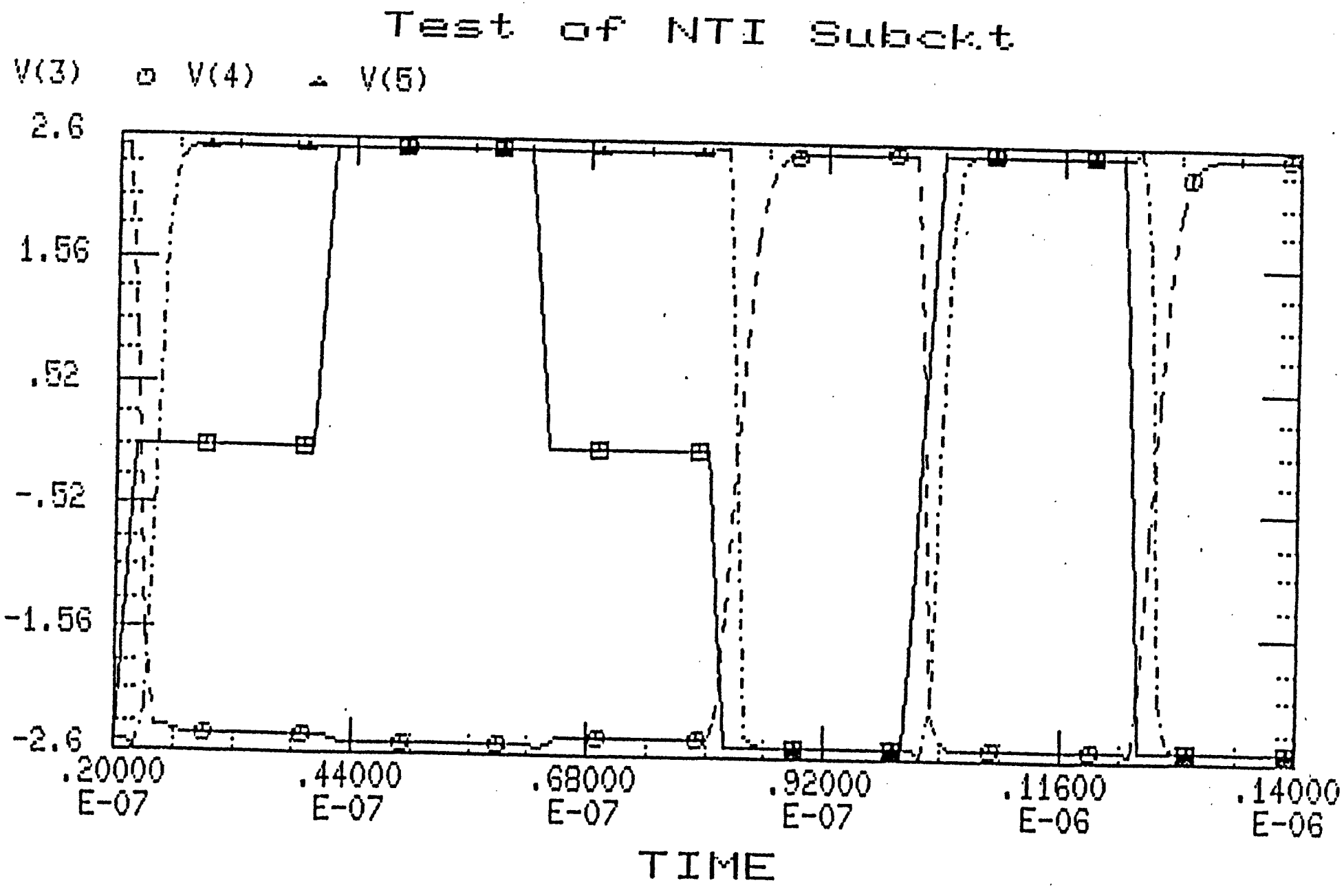
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of NTI Subckt
*^^^^^^^^^^^^^^^^^^^^^^^^
* Start of Subcircuits
*^^^^^^^^^^^^^^^^^^^^^^^^
*--------------------------
* Negative ternary inverter
*--------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT NTI 1 2 3 4
M1 4 3 1 1 PMOSE W=7.2U  L=3.0U AD=50P  AS=75P  PD=30U PS=38U NRD=1.3 NRS=2.0
M2 4 3 2 2 NMOSE W=21.6U L=3.0U AD=149P AS=225P PD=59U PS=67U NRD=0.4 NRS=0.7
.ENDS NTI
*-------------------------------------
* Test Power Supply for Ternary Circuits
*-------------------------------------
* 1=OUTPUT
.SUBCKT SOURCE 1
VIN1 1 2 PULSE( 0  2.5 40NS 2NS 2NS 20NS  60NS)
VIN2 2 3 PULSE(-2.5  0 20NS 2NS 2NS 100NS 120NS)
VIN3 3 0 PULSE( 0 -2.5 80NS 2NS 2NS 20NS  120NS)
.ENDS SOURCE
*^^^^^^^^^^^^^^^^^^^^
* Start of Circuits
*^^^^^^^^^^^^^^^^^^^^
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5

X1 1 2 3 4 NTI
X2 1 2 4 5 NTI
XSRC 3 SOURCE

*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* Start of Output Generation
*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
.TRAN 1NS 140NS 20NS
'.PRINT TRAN V(3) V(4) V(5)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

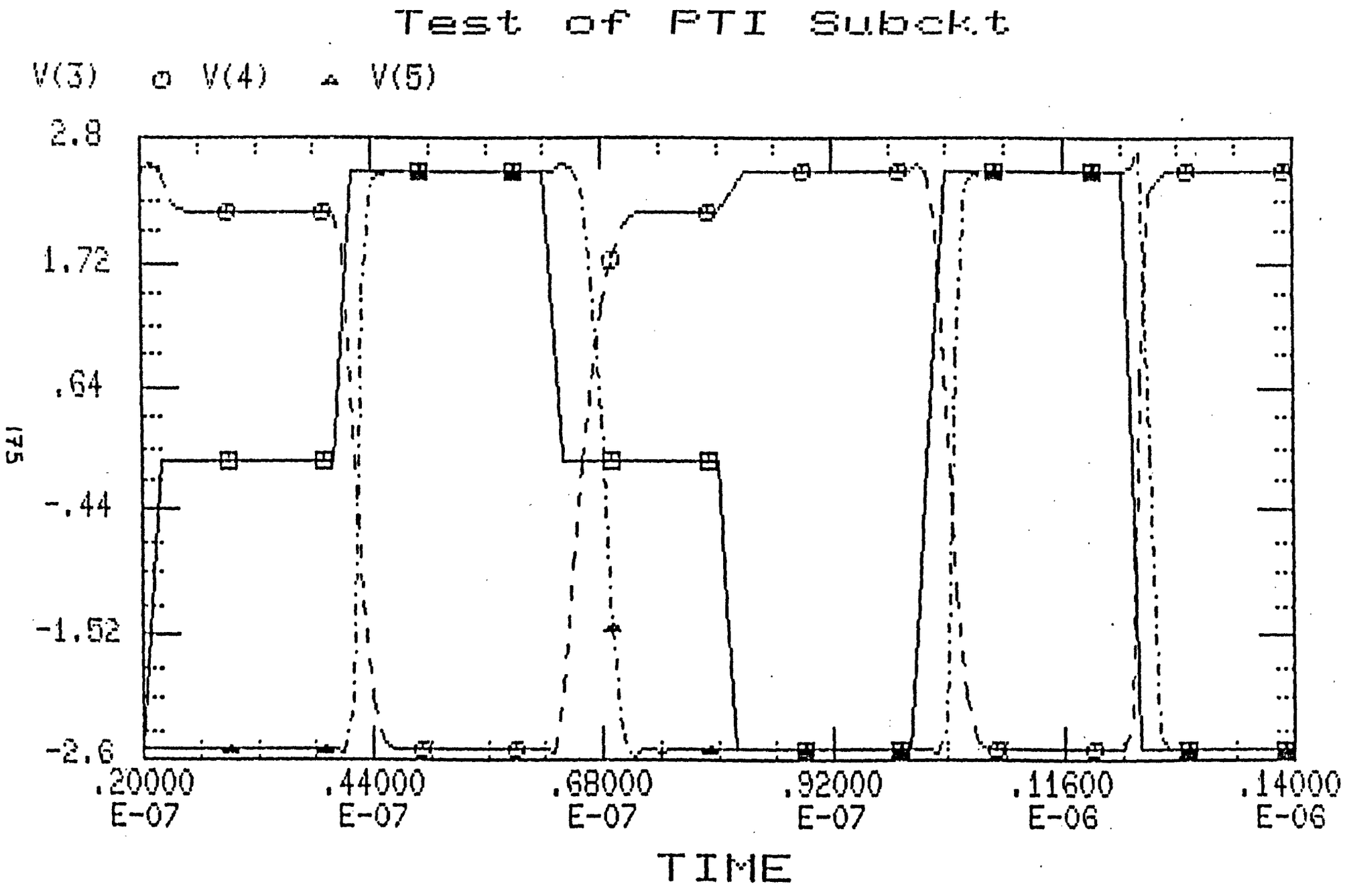
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of PTI Subckt
*********************
* Start of Subcircuits
*********************
*---------------------------
* Positive ternary inverter
*---------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT PTI 1 2 3 4
M1 4 3 1 1 PMOSE W=24U L=3U PD=62U PS=62U AD=187P AS=56P NRD=0.5 NRS=0.3
M2 4 3 2 2 NMOSE W=3U  L=3U PD=26U PS=35U AD=37P  AS=60P NRD=3.6 NRS=4.0
.ENDS PTI
*---------------------------------------
* Test Power Supply for Ternary Circuits
*---------------------------------------
* 1=OUTPUT
.SUBCKT SOURCE 1
VIN1 1 2 PULSE( 0  2.5 40NS 2NS 2NS 20NS  60NS)
VIN2 2 3 PULSE(-2.5  0 20NS 2NS 2NS 100NS 120NS)
VIN3 3 0 PULSE( 0 -2.5 80NS 2NS 2NS 20NS  120NS)
.ENDS SOURCE

*********************
* Start of Circuits
*********************
VCC 1 0 DC 2.5
VSS 2 0.DC -2.5

X1 1 2 3 4 PTI
X2 1 2 4 5 PTI
XSRC 3 SOURCE

*****************************
* Start of Output Generation
*****************************
.TRAN 1NS 140NS 20NS
.PRINT TRAN V(3) V(4) V(5)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

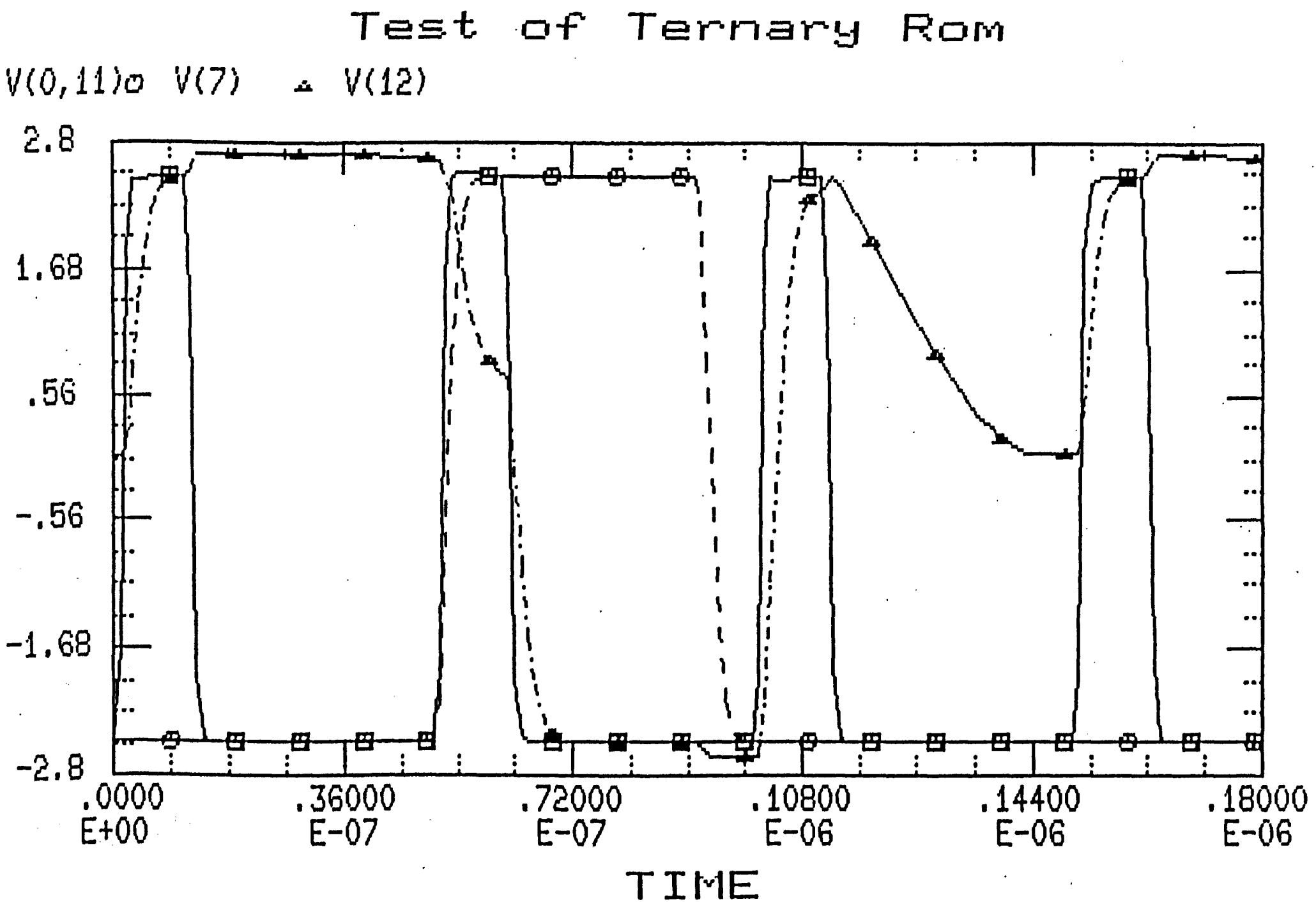
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



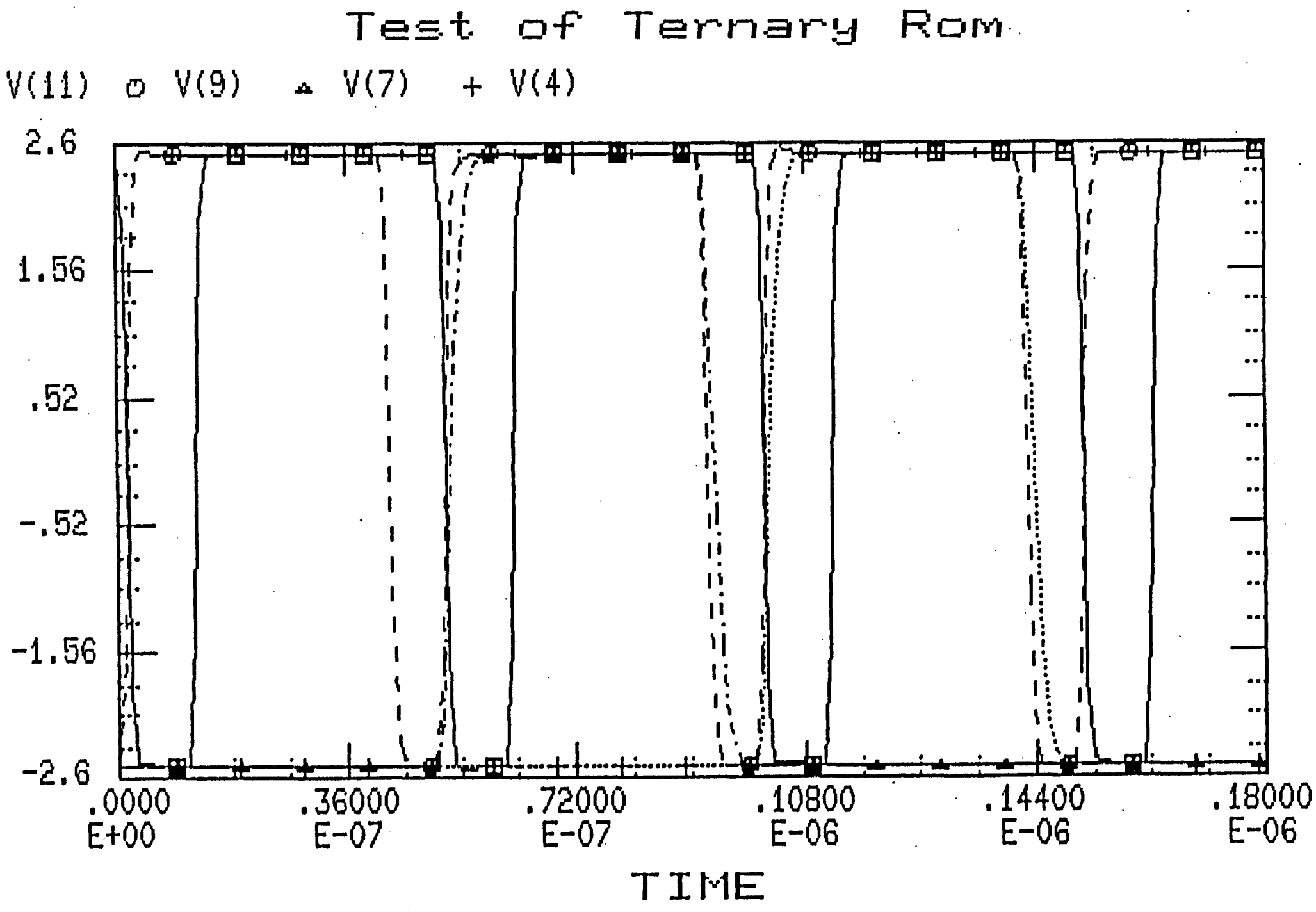
```
Test of Ternary Rom
*"""""""""""""""""""""""
* Start of Subcircuits
*"""""""""""""""""""""""
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV

*"""""""""""""""""""
* Start of Circuits
*"""""""""""""""""""
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5

M1 12 11 1 1 PMOSE W=10.8U L=3.0U AD=91P AS=71P PD=38U PS=35U NRD=0.8 NRS=0.6
M2 12  9 5 2 NMOSE W=5.4U L=3.0U AD=45P AS=45P PD=28U PS=28U NRD=1.6 NRS=1.6
M3  5  7 2 2 NMOSE W=5.4U L=3.0U AD=24P AS=16P PD=20U PS=17U NRD=0.8 NRS=0.6
M4  5  4 0 2 NMOSE W=5.4U L=3.0U AD=24P AS=16P PD=20U PS=17U NRD=0.8 NRS=0.6
X1 1 2 10 11 RATINV
X2 1 2 8 9 RATINV
X3 1 2 6 7 RATINV
X4 1 2 3 4 RATINV
C1 9 1 20FF
C2 9 2 5.6FF
C3 7 2 108FF
C4 4 2 108FF
C5 12 1 176FF
C6 5 2 32FF

VPR 10 0 PULSE(-2.5 2.5 0 0 0 10NS 50NS)
VCS  8 0 PULSE(2.5 -2.5 0 0 0 40NS 50NS)
VRS1 6 0 PULSE(2.5 -2.5 50NS 0 0 40NS 150NS)
VRS2 3 0 PULSE(2.5 -2.5 100NS 0 0 40NS 150NS)
*"""""""""""""""""""""""""""
* Start of Output Generation
*"""""""""""""""""""""""""""
.NODESET V(5)=2.5 V(12)=2.5 V(9)=0
.TRAN 1NS 180NS
.PRINT TRAN V(11) V(9) V(7) V(4)
.PRINT TRAN V(0,11) V(7) V(12)
.PRINT TRAN V(0,11) V(4) V(12)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of Ternary Latch System
*^^^^^^^^^^^^^^^^^^^^^^^^
* Start of Subcircuits
*^^^^^^^^^^^^^^^^^^^^^^^^
*---------------------------------------
* Test Power Supply for Ternary Circuits
*---------------------------------------
* 1=VA 2=VB 3=V3
.SUBCKT SOURCE 1 3 2
VA  1 0 PULSE(2.5 -2.5 80NS 0 0 40NS 120NS)
VB1 2 4 PULSE(0 5 0 0 0 40NS 240NS)
VB2 4 0 PULSE(-2.5 2.5 160NS 0 0 40NS 240NS)
VC1 3 5 PULSE(0 5 40NS 0 0 40NS 240NS)
VC2 5 0 PULSE(-2.5 2.5 120NS 0 0 40NS 240NS)
.ENDS SOURCE
*--------------------------
* Positive ternary inverter
*--------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT PTI 1 2 3 4
M1 4 3 1 1 PMOSE W=24U L=3U PD=62U PS=62U AD=187P AS=56P NRD=0.5 NRS=0.3
M2 4 3 2 2 NMOSE W=3U  L=3U PD=26U PS=35U AD=37P  AS=60P NRD=2.6 NRS=4.0
.ENDS PTI
*--------------------------
* Negative ternary inverter
*--------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT NTI 1 2 3 4
M1 4 3 1 1 PMOSE W=3U   L=3U PD=26U PS=35U AD=36P  AS=60P NRD=2.6 NRS=4.0
M2 4 3 2 2 NMOSE W=5.4U L=3U PD=25U PS=57U AD=167P AS=62P NRD=1.3 NRS=2.1
.ENDS NTI
*------------------
* 2 input Nand gate
*------------------
* 1=VCC 2=VSS 3=A 4=B 5=OUTPUT
.SUBCKT NAND 1 2 3 4 5
M1 5 3 1 1 PMOSE W=5.4U L=3.0U AD=28P AS=65P PD=21U PS=35U NRD=0.9 NRS=2.2
M2 5 4 1 1 PMOSE W=5.4U L=3.0U AD=28P AS=42P PD=21U PS=26U NRD=0.9 NRS=1.4
M3 5 4 6 2 NMOSE W=5.4U L=3.0U AD=42P AS=8P  PD=26U PS=14U NRD=1.4 NRS=0.3
M4 6 3 2 2 NMOSE W=5.4U L=3.0U AD=8P  AS=65P PD=14U PS=35U NRD=0.3 NRS=2.2
.ENDS NAND
*-----------------
* Ratioed Inverter
*-----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*---------------
* Ternary 3 to 1
*---------------
* 1=VCC 2=VSS 3=Ain 4=Bin 5=Cin 6=Vout
.SUBCKT T3TO1 1 2 3 4 5 6
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

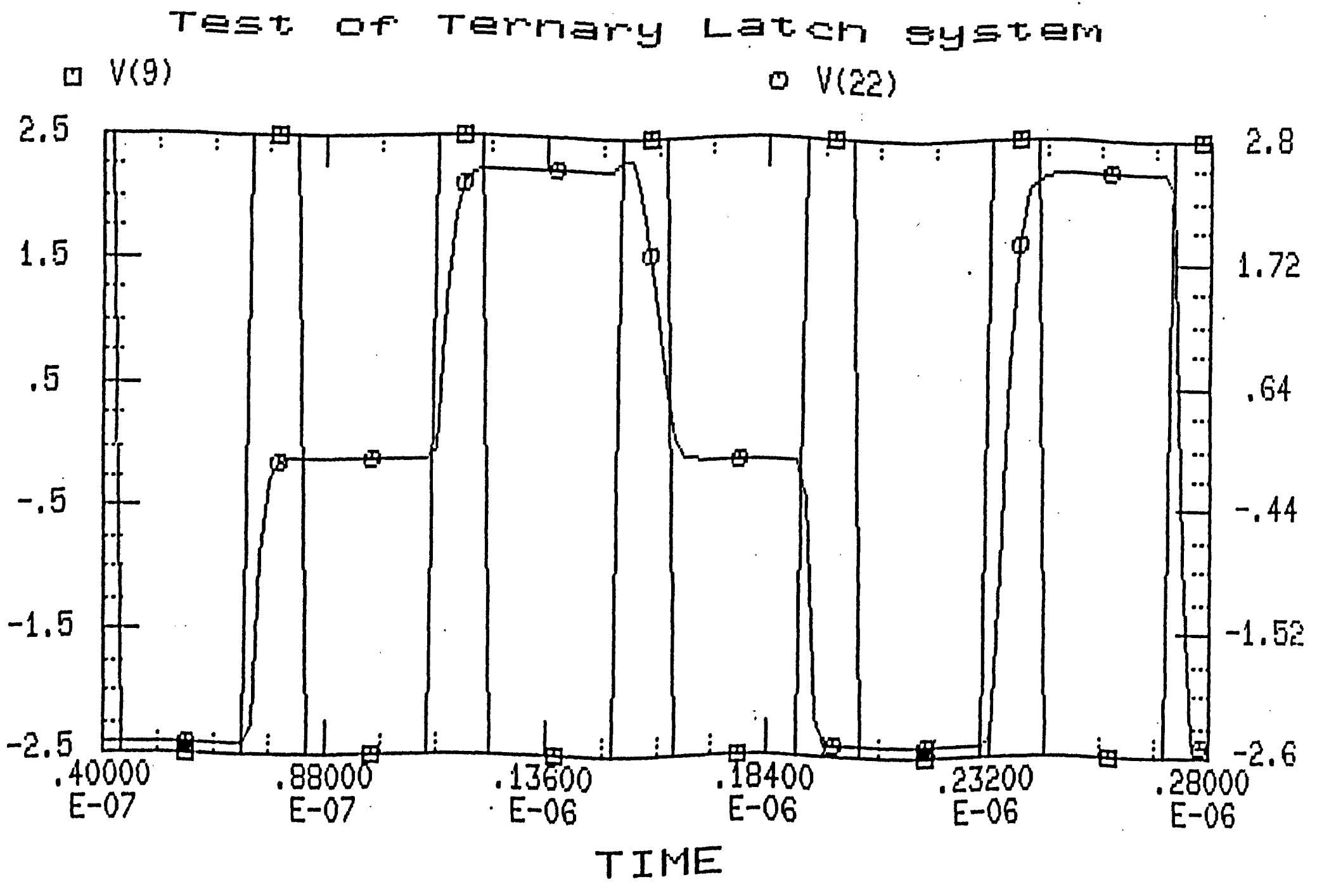
```
M1 6 3 1 1 PMOSE W=5.4U L=3.0U AD=42P AS=65P PD=26U PS=35U NRD=1.4 NRS=2.2
M2 6 4 0 2 NMOSE W=5.4U L=3.0U AD=28P AS=65P PD=21U PS=35U NRD=0.9 NRS=2.2
M3 6 5 2 2 NMOSE W=5.4U L=3.0U AD=28P AS=42P PD=21U PS=26U NRD=0.9 NRS=1.4
.ENDS T3TO1
*------------------
* TRANSMISSION GATE
*------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT 5=CONT 6=CONT~
.SUBCKT TGATE 1 2 3 4 5 6
M1 3 6 4 1 PMOSE W=5.4U L=3.0U AD=45P AS=45P PD=28U PS=28U NRD=1.6 NRS=1.6
M2 3 5 4 2 NMOSE W=5.4U L=3.0U AD=45P AS=45P PD=28U PS=28U NRD=1.6 NRS=1.6
.ENDS TGATE
*~~~~~~~~~~~~~~~~~~~
* Start of Circuits
*~~~~~~~~~~~~~~~~~~~
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5

* 4=VSS 1=VA 2=VB 3=V3
X1 3 4 5 SOURCE

* 1=VCC 2=VSS 3=Ain 4=Bin 5=Cin 6=Vout
X2 1 2 3 4 5 6 T3TO1
X3 1 2 19 20 21 22 T3TO1

* 1=VCC 2=VSS 3=INPUT 4=OUTPUT 5=CONT 6=CONT~
X4 1 2 6 11 7 8 TGATE
X5 1 2 13 19 9 10 TGATE
X6 1 2 17 20 9 10 TGATE
X7 1 2 18 21 9 10 TGATE

X8 1 2 11 13 PTI
X9 1 2 11 12 NTI
X10 1 2 13 15 RATINV
X11 1 2 12 14 RATINV
X12 1 2 14 18 RATINV
X13 1 2 16 17 RATINV
X14 1 2 13 14 16 NAND

C1 11 0 50FF
C2 13 0 30FF
C3 17 0 30FF
C4 18 0 30FF
C5 19 0 10FF
C6 20 0 10FF
C7 21 0 10FF
C8 22 0 100FF

VPH1  7 0 PULSE(-2.5 2.5 5NS 0 0 20NS 40NS)
VPH1B 8 0 PULSE(2.5 -2.5 5NS 0 0 20NS 40NS)
VPH2  9 0 PULSE(-2.5 2.5 30NS 0 0 10NS 40NS)
VPH2B 10 0 PULSE(2.5 -2.5 30NS 0 0 10NS 40NS)
*~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Start of Output Generation
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

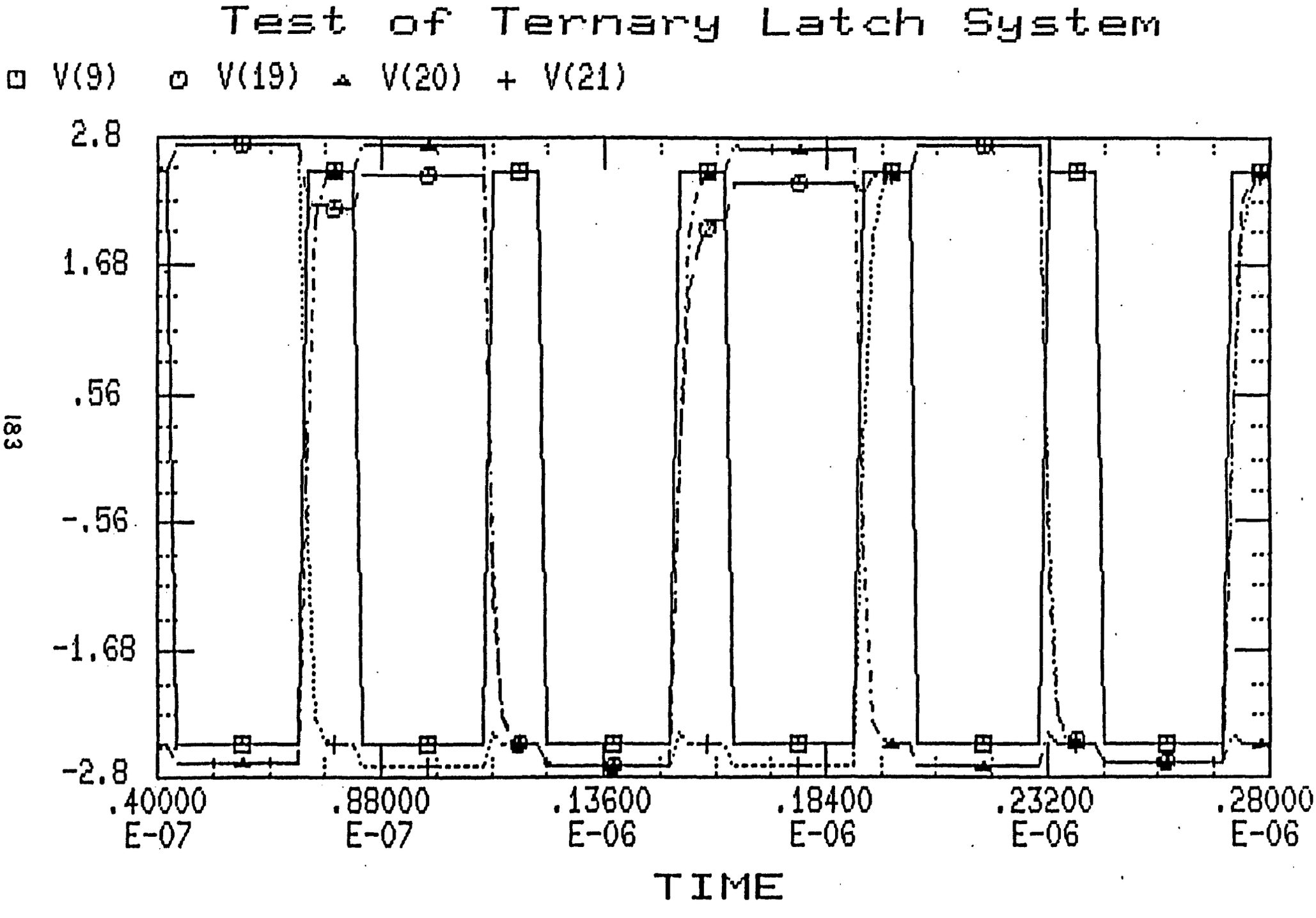
```
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.TRAN 2NS 280NS 40NS
.PRINT TRAN V(7) V(11) V(13) V(12)
.PRINT TRAN V(9) V(19) V(20) V(21)
.PRINT TRAN V(9) V(22)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

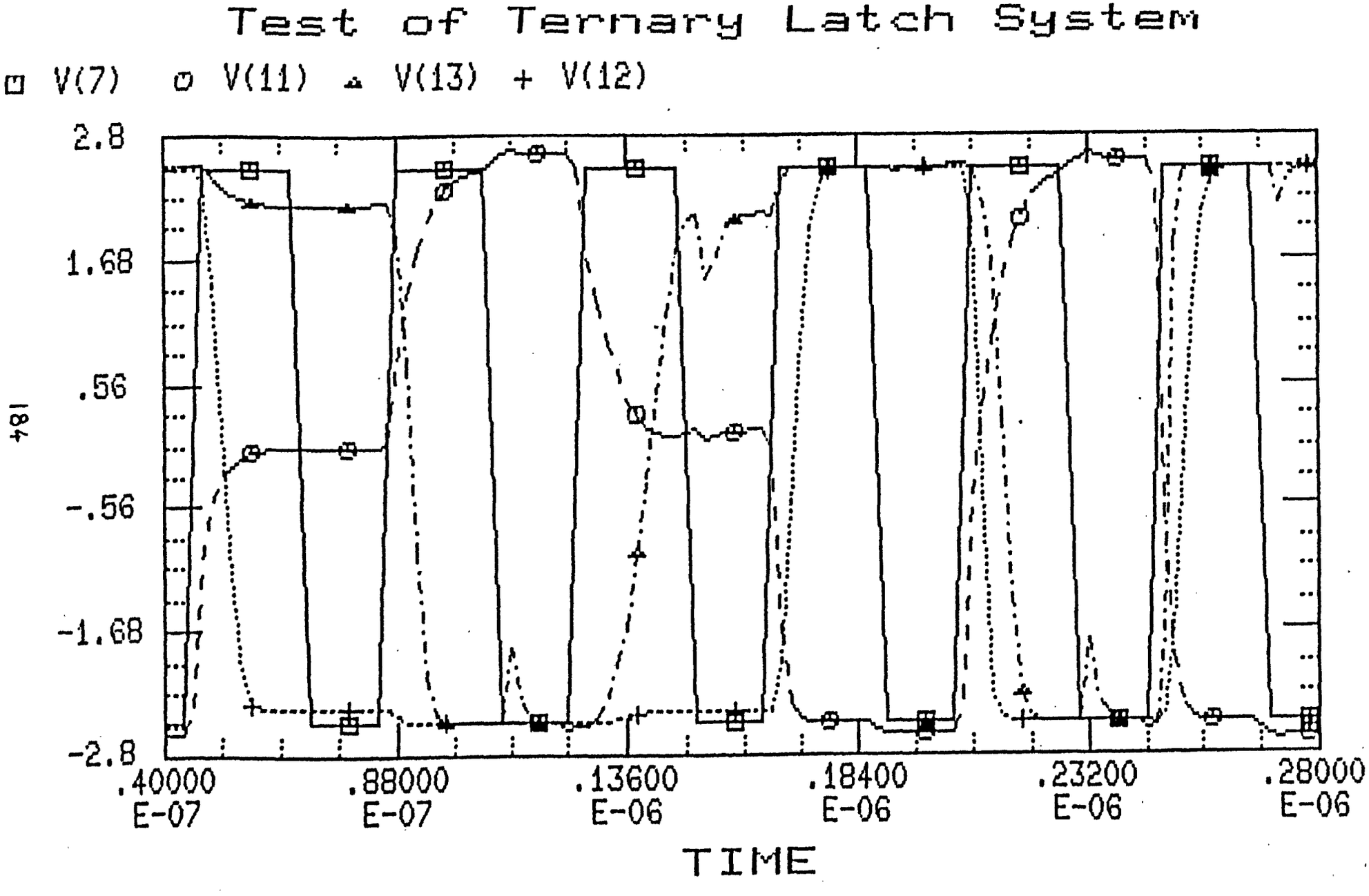
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



```
Test of Ternary Row Decoder
*++++++++++++++++++++++
* Start of Subcircuits
*++++++++++++++++++++++
*---------------------------------------
* Test Power Supply for Ternary Circuits
*---------------------------------------
* 1=VA 2=VB 3=V3
.SUBCKT SOURCE 1 3 2
VA  1 0 PULSE(2.5 -2.5 80NS 0 0 40NS 120NS)
VB1 2 4 PULSE(0 5 0 0 0 40NS 240NS)
VB2 4 0 PULSE(-2.5 2.5 160NS 0 0 40NS 240NS)
VC1 3 5 PULSE(0 5 40NS 0 0 40NS 240NS)
VC2 5 0 PULSE(-2.5 2.5 120NS 0 0 40NS 240NS)
.ENDS SOURCE
*-------------------------
* Positive ternary inverter
*-------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT PTI 1 2 3 4
M1 4 3 1 1 PMOSE W=24U L=3U PD=62U PS=62U AD=187P AS=56P NRD=0.5 NRS=0.3
M2 4 3 2 2 NMOSE W=3U  L=3U PD=26U PS=35U AD=37P  AS=60P NRD=2.6 NRS=4.0
.ENDS PTI
*-------------------------
* Negative ternary inverter
*-------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT NTI 1 2 3 4
M1 4 3 1 1 PMOSE W=3U   L=3U PD=26U PS=35U AD=36P  AS=60P NRD=2.6 NRS=4.0
M2 4 3 2 2 NMOSE W=5.4U L=3U PD=25U PS=57U AD=167P AS=62P NRD=1.3 NRS=2.1
.ENDS NTI
*-----------------
* 2 input Nand gate
*-----------------
* 1=VCC 2=VSS 3=A 4=B 5=OUTPUT
.SUBCKT NAND 1 2 3 4 5
M1 5 3 1 1 PMOSE W=5.4U L=3.0U AD=28P AS=65P PD=21U PS=35U NRD=0.9 NRS=2.2
M2 5 4 1 1 PMOSE W=5.4U L=3.0U AD=28P AS=42P PD=21U PS=26U NRD=0.9 NRS=1.4
M3 5 4 6 2 NMOSE W=5.4U L=3.0U AD=42P AS=8P  PD=26U PS=14U NRD=1.4 NRS=0.3
M4 6 3 2 2 NMOSE W=5.4U L=3.0U AD=8P  AS=65P PD=14U PS=35U NRD=0.3 NRS=2.2
.ENDS NAND
*----------------
* Ratioed Inverter
*----------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT
.SUBCKT RATINV 1 2 3 4
M1 4 3 1 1 PMOSE W=9.3U L=3.0U AD=51P AS=74P PD=34U PS=43U NRD=1.4 NRS=2.2
M2 4 3 2 2 NMOSE W=3.0U L=3.0U AD=36P AS=59P PD=26U PS=35U NRD=2.6 NRS=4.0
.ENDS RATINV
*--------------
* Ternary 3 to 1
*--------------
* 1=VCC 2=VSS 3=Ain 4=Bin 5=Cin 6=Vout
.SUBCKT T3TO1 1 2 3 4 5 6
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

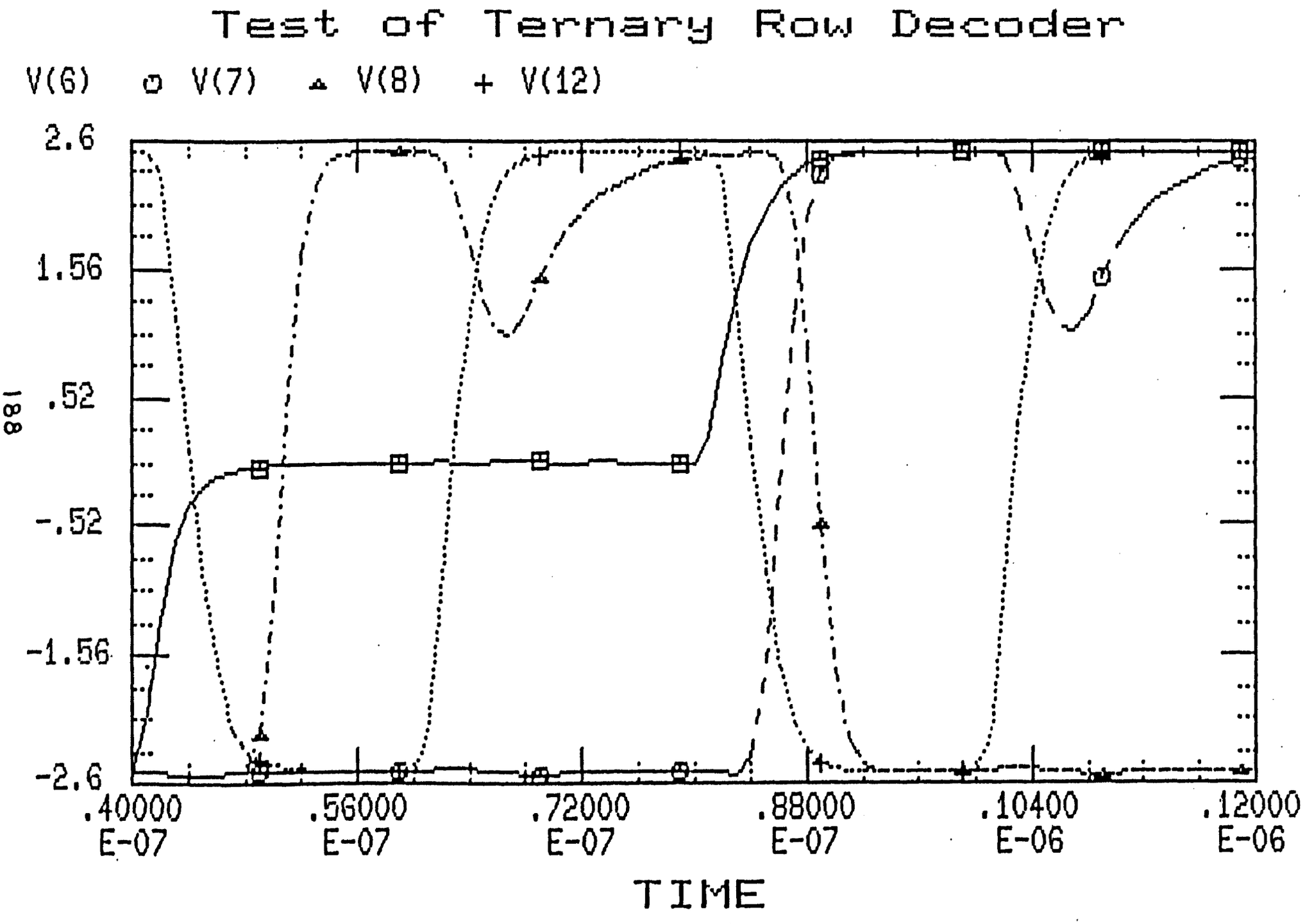
```
M1 6 3 1 1 PMOSE W=5.4U L=3.0U AD=42P AS=65P PD=26U PS=35U NRD=1.4 NRS=2.2
M2 6 4 0 2 NMOSE W=5.4U L=3.0U AD=28P AS=65P PD=21U PS=35U NRD=0.9 NRS=2.2
M3 6 5 2 2 NMOSE W=5.4U L=3.0U AD=28P AS=42P PD=21U PS=26U NRD=0.9 NRS=1.4
.ENDS T3TO1
*---------------
* Decoder 1 to 3
*---------------
* 1=VCC 2=VSS 3=Input 4=C2 5=C1 6=C0
.SUBCKT DEC1TO3 1 2 3 8 9 10
X8 1 2 3 4 PTI
X9 1 2 3 5 NTI
X10 1 2 4 8 RATINV
X11 1 2 5 6 RATINV
X12 1 2 6 10 RATINV
X13 1 2 7 9 RATINV
X14 1 2 4 6 7 NAND
C1 3 1 50FF
.ENDS DEC1TO3
*------------------------------
* transmission gate matrix cell
*------------------------------
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT 5=CONT 6=CONT~
.SUBCKT TGMAT 1 2 3 4 5 6
M1 3 6 4 1 PMOSE W=5.4U L=3.0U AD=45P AS=45P PD=28U PS=28U NRD=1.6 NRS=1.6
M2 3 5 4 2 NMOSE W=5.4U L=3.0U AD=45P AS=45P PD=28U PS=28U NRD=1.6 NRS=1.6
M3 4 6 2 2 NMOSE W=5.4U L=3.0U AD=45P AS=45P PD=28U PS=28U NRD=1.6 NRS=1.6
.ENDS TGMAT
*~~~~~~~~~~~~~~~~~~~
* Start of Circuits
*~~~~~~~~~~~~~~~~~~~
VCC 1 0 DC 2.5
VSS 2 0 DC -2.5

* 1=VA 2=VB 3=V3
X1 3 4 5 SOURCE
* 1=VCC 2=VSS 3=Ain 4=Bin 5=Cin 6=Vout
X2 1 2 3 4 5 6 T3TO1
X3 1 2 6 7 8 9 DEC1TO3
* 1=VCC 2=VSS 3=INPUT 4=OUTPUT 5=CONT 6=CONT~
X4 1 2 8 10 12 13 TGMAT
X5 1 2 7 11 12 13 TGMAT
X6 1 2 14 12 RATINV
X7 1 2 12 13 RATINV
C1 8 1 67FF
C2 7 1 67FF
C3 10 1 113FF
C4 11 1 113FF
C5 13 1 129FF
C6 12 1 105FF

VCS 14 0 PULSE(2.5 -2.5 20NS 0 0 20NS 40NS)
*~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Start of Output Generation
*~~~~~~~~~~~~~~~~~~~~~~~~~~~
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

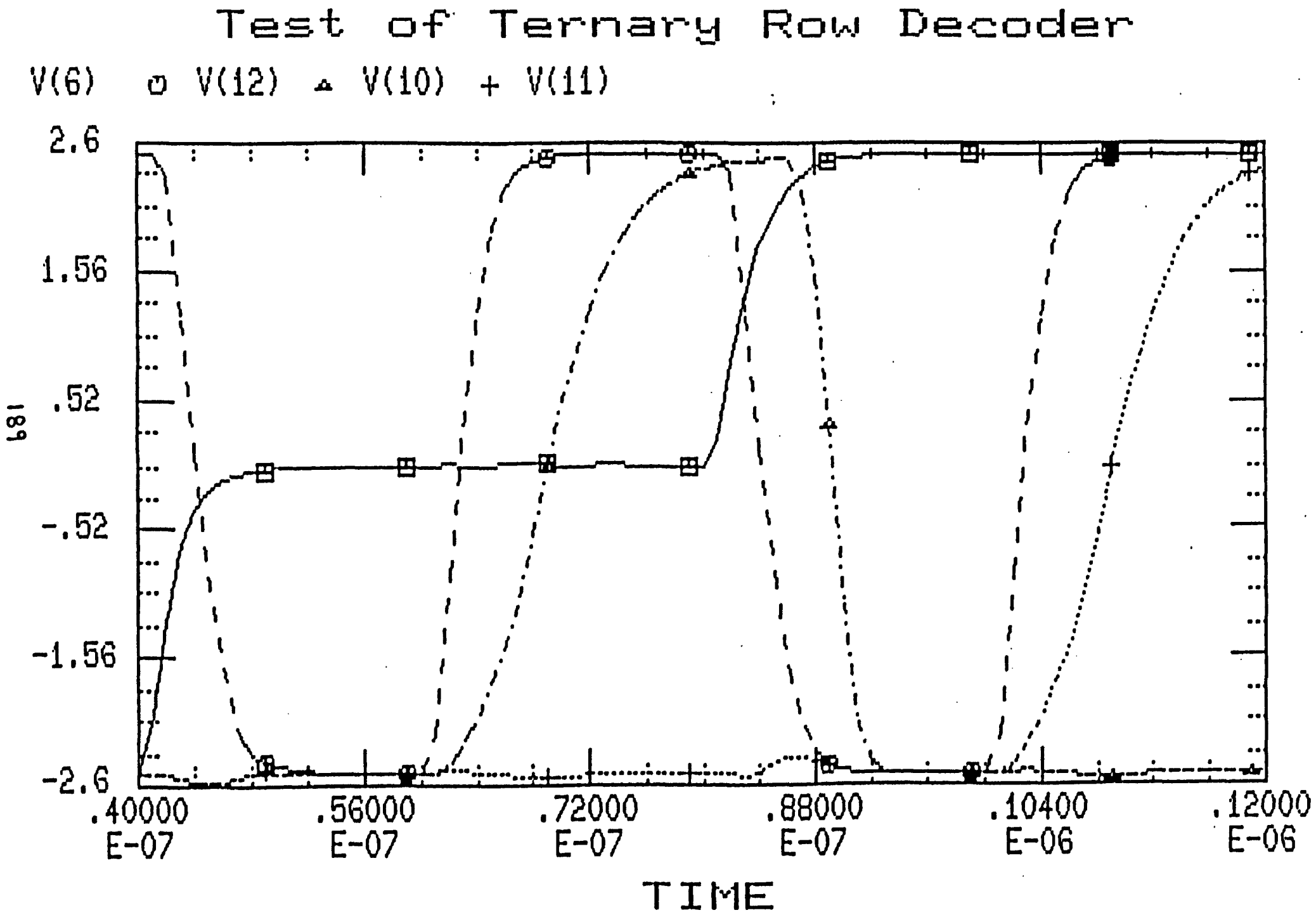
```
.TRAN 1NS 120NS 40NS
.PRINT TRAN V(6) V(7) V(8) V(12)
.PRINT TRAN V(6) V(12) V(10) V(11)
.END
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Appendix V

Silicon Compiler and Centering Program

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix V. Silicon Compiler and Centering Program

The operation of the Silicon Compiler program has already been discussed in section IV and nothing further needs to be said about it.

The centering program accepts as input, a file containing the CIF code for a design that is to be checked. In the first pass through the CIF code, the program interprets all the commands in order to calculate the extents of the individual cells. During this pass, the program also monitors the vertices to make sure that they are multiples of 50. There are three other items that are check because they could have an effect on whether or not a design is likely to be accepted for submission. It is required by CMC, the Canadian Micro Electronic Corporation, that all designs have fewer than five levels, and that the number of 12 and 20 point polygons be less than 3501 and 2501 respectively.

If the program finds that some cells are not centered or that there are vertices that are not multiples of 50, then the program makes another pass through the CIF data file. This time, a new data file is created with the CIF data modified to center the cells and snap all vertices to multiples of 50. It is useful to be able to run designs through this program before they are sent of to CMC for fabrication.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
C-----------------------------------------------------------------------
C This program generates 5 data files, rom0 through rom5, which
C contain the cif code used to program the roms for the new
C binary design.
C-----------------------------------------------------------------------
      implicit integer (a-z)
      character*40 fname
      integer colad(0:3)/8,0,16,24/,rowad(0:7)/1,0,2,3,5,4,6,7/
      integer bitpos(0:4)/4,0,1,2,3/
      integer dely(0:7)/8*22/,delx(0:3)/4*94/

      xstart=0
      ystart=0
      modulus=22
      dxbit = 16

      do rom=0,4
          fname = 'rom'//char(48+rom)//'.cif'
          open(unit=1,status='unknown',file=fname,carriagecontrol='list')
          write(1,'(a)') 'DS 1;'
          write(1,'(a)') '( Begin symbol: TRAN );'
          write(1,'(a)') 'L CF;'
          write(1,'(a)') 'B 900 1300 0 0;'
          write(1,'(a)') 'DF;'
          write(1,'(a)') '( End symbol: TRAN );'
          write(1,'(a,il,a)') '( Begin symbol: ROM',rom,' );'

          xpos=xstart
          ypos=ystart
          do col = 0,3
              do row = 0,7
                  write(*,'(a,il,a,il,a,il,a)') char(0)//
     1                                          'rom=',rom,' col=',col,
     1                                          ' row=',row,char(13)
                  address = colad(col) + rowad(row)
                  data = jmod(address + 2**rom,modulus)
                  do bit=0,4
                      tran = jibits(data,bit,1)
                      if (tran.eq.1) then
                          write(1,'(a,i6,a,i6,a)')
     1                    'C1 T',(xpos+dxbit*bitpos(bit))*100,
     1                    ',',ypos*100,';'
                      endif
                  enddo
                  ypos = ypos + dely(row)
              enddo
              xpos = xpos + delx(col)
              ypos = ystart
          enddo
          write(1,'(a,il,a)') ' ( End symbol: ROM',rom,' );'
          write(1,*) 'E'
          close(unit=1)
      enddo
      end
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
C-----------------------------------------------------------------
C This program examines a CIF data file and produces statistics on
C all of the cells contained in the design.  It also centers any
C cells uncentered.  The program snaps all grid coordinates to
C multiples of 50 units.  Written by: Peter Bird, January 1987
C-----------------------------------------------------------------
      implicit integer(c,t,d,s)
      include 'global.hdr'
      include 'cursor1.hdr'

      character*80 error_msg,tline,line,ch*1,fill*10
      character cell_name(0:99)*10/100*' '/
      integer stack(0:5000),sp/-1/,points(20),num_pts
      integer num_12_pt,num_20_pt,levels_below(0:99)/100*0/
      integer num12pt(0:99)/100*0/,num20pt(0:99)/100*0/
      integer line_num/0/,movex(0:99),movey(0:99)
      integer maxx(0:99)/100* -5000000/,maxy(0:99)/100* -5000000/
      integer minx(0:99)/100*5000000/  ,miny(0:99)/100*5000000/
      integer num_calls(0:99)/100*0/,cn,vert,horiz,screen/27/
      integer mout(4),minp(4),mtemp(4)
      integer mirrorx(4,4)/0,0,-1,0,0,1,0,0,-1,0,0,0,0,0,0,1/
      integer mirrory(4,4)/1,0,0,0,0,0,0,-1,0,0,1,0,0,-1,0,0/
      integer rot__01(4,4)/0,1,0,0,0,0,-1,0,0,0,0,1,-1,0,0,0/
      integer rot_m10(4,4)/0,0,-1,0,0,0,0,-1,-1,0,0,0,0,-1,0,0/
      integer rot_0m1(4,4)/0,0,0,-1,1,0,0,0,0,-1,0,0,0,0,1,0/
      integer del_mirx(4,4)/-1,0,2*0,0,1,10*0/
      integer del_miry(4,4)/1,0,2*0,0,-1,10*0/
      integer del__r01(4,4)/0,1,2*0,-1,0,10*0/
      integer del_rm10(4,4)/-1,0,2*0,0,-1,10*0/
      integer del_r0m1(4,4)/0,-1,2*0,1,0,10*0/
      logical too_many_levels/.false./,centered/.true./
      logical multiple_of_50/.true./
      logical in_symbol,stack_empty,need_line/.true./,no_semic
      common /stack/stack,sp,stack_empty
      common /info/cell_name,maxx,maxy,minx,miny,num_calls,cn,line_num
      common /error/error_msg,no_semic,multiple_of_50
      m50(i)= sign((abs(i)+25)/50*50,i)
      include 'cursor2.hdr'

c*****************************************************************
c variables:
c   maxx,maxy,minx,miny = coordinates of corners of bounding box of cell
c   cell_name = name of cell from Daisy
c   line_num  = current line of cif file being processed
c   stack     = after a cell is defined data is pushed onto this stack
c                in the form:
c     (-dscell#-100)  cell# count [cell# count]... (-dscell#-100)
c           where dscell# = cell number of cell just defined
c                 cell# = cell number called
c                 count = number of times cell was called
c   sp        = stack pointer
c   error_msg = a string indicating the nature of the error
c   num_calls = during the program it is used to tabulate how many times
c               each cell is called, just before the report is produced
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c               the total of all calls is computed using the above stacks,
c               this array holds the results.
c   line  = a line of input from the cif data file
c   tline = partial line used in parsing the input line
c   in_symbol = true if currently processing a symbol - can't define
c               another symbol if this variable is true
c   need_line - a new line is read only if this is true.  The only time
c               need_line is false occurs when the line after a DS
c               statement does not contain the symbol name
c   no_semic  = .true. if no semicolon is found in a line
c   movex,movey = amount the cell coordinates have to be moved for the
c                 cell to be centered
c   levels_below = how many levels are below the current cell.  If a cell
c                  calls another cell then there is a level below it.
c   points = sequence of numbers found in line being parsed
c   num_12_pt,num_20_pt = number of polygons with more than 12 and 20 points
c   num_pts = number of points found when line was parsed
c   m50 takes an integer and makes it a multiple of 50.
c         25-74 -> 50, 75-124 -> 100 ...etc.
c   mout,mtemp,minp are matrices used in the rotate and mirror commands
c        they represent output,output & input respectively
c   the 4x4 matrices contain the data that perform the actual rotation
c   and mirroring operations.
c*****************************************************************************
        call getfile('I',finl,'Enter the cif filename: ',stat)
        if (.not.stat) stop
        open(unit=3,status='UNKNOWN',file='center.rpt',carriagecontrol='list')
        open(unit=2,status='UNKNOWN',file='center.msg',carriagecontrol='list')
        open(unit=1,status='UNKNOWN',file=finl,carriagecontrol='list')

        cell_name(0)='Top level'
        call tcom(cls)
        call print_at(10,screen,'Processing Cell:')

c------------------------
c start of main read loop
c------------------------
        do while(.true.)        ! loop terminated by return(1)
                                ! inside getline -> to line 10
            if (need_line) then
                call getline(line,line_num,length,&10)
                ch=line(1:1)
                if (no_semic .and. ch.ne.'P' .and. ch.ne.'E') goto 99
            endif
            need_line=.true.

c-----------------------------------------------
c Define Start - get symbol number and cell name
c-----------------------------------------------
            if (line(1:2).eq.'DS') then
                if (in_symbol) then
                    error_msg = 'can not define a symbol inside another symbol'
                    goto 99
                endif
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                in_symbol = .true.
                read(line,'(3x,i<index(line,'';'')-4>)') cn
                call push(-cn-100,&99)

                call getline(line,line_num,length,&10)
                if (line(1:7).eq.'( Begin') then
                   cell_name(cn) = line(17:index(line,';')-3)
                else
                    i=2
                    if (i.le.9) i=1
                    write(cell_name(cn),'(a,i<i>)') 'Cell_',cn
                    need_line=.false.
                endif
                call print_at(10,screen+17,cell_name(cn))
                call tcom(cteol)

c-------------------------------------------------
c Define Finish - restore cell number to top level
c-------------------------------------------------
            else if (line(1:2).eq.'DF') then
                if (.not.in_symbol) then
                    error_msg = 'DF without matching DS'
                    goto 99
                endif
                in_symbol = .false.
                do i=1,99
                    if (num_calls(i).gt.0) then
                        call push(i,&99)
                        call push(num_calls(i),&99)
                        num_calls(i)=0
                    endif
                enddo
                call push(-cn-100,&99)
                cn = 0
                call print_at(10,screen+17,cell_name(cn))
                call tcom(cteol)

c-----------------------------------------
c Parse box statements
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c format: B horiz vert cenx ceny;
c where horiz = horizontal size
c       vert  = verical size
c       cenx  = x coordinate of center of box
c       ceny  = y coordinate of center of box
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
            else if (line(1:1).eq.'B') then

                if (.not.in_symbol) then   ! when all cells have been defined
                        call push(-100,&99)! and one returns to the top level,
                        in_symbol=.true.   ! in_symbol=.f. Now push -100 to
                endif                      ! indicate this is the top level

                call parse(line(1:length),length,points,np,&99)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                call check(line(1:length),length,points,np)
                horiz= points(1)
                vert = points(2)
                cenx = points(3)
                ceny = points(4)
                if (np.ne.4 .or. horiz.le.0 .or. vert.le.0) then
                    error_msg='invalid box command'
                    goto 99
                endif
                call update(cn,cenx+horiz/2,ceny+vert/2,
       1                        cenx-horiz/2,ceny-vert/2)

c-------------------------------------
c Parse polygon statements
c*************************************
c format: P x1,y1 x2,y2 ...
c         xi,yi ...
c         xj,yj ... xn,yn;
c where x,y are the coordinates of the
c       vertices of the polygon
c*************************************
            else if (line(1:1).eq.'P') then

                if (.not.in_symbol) then   ! when all cells have been defined
                        call push(-100,&99)! and one returns to the top level,
                        in_symbol=.true.   ! in_symbol=.f. Now push -100 to
                endif                      ! indicate this is the top level

                done=.false.
                num_pts=0
                do while(.not.done)
                    call parse(line(1:length),length,points,np,&99)
                    call check(line(1:length),length,points,np)
                    num_pts=num_pts+np
                    do i=1,np,2
                        ix=points(i)
                        iy=points(i+1)
                        call update(cn,ix,iy,ix,iy)
                    enddo
                    i=index(line,';')
                    if (i.gt.0 .and. i.le.length) then
                        done=.true.
                    else
                        call getline(line,line_num,length,&97)
                        ch=line(1:1)
                        if (ch.eq.'(' .or. ch.eq.'D' .or. ch.eq.'P' .or.
       1                    ch.eq.'C' .or. ch.eq.'B' .or. ch.eq.'E') then
                                error_msg = 'missing semicolon in previous line'
                                goto 99
                        endif
                    endif
                enddo
                if (num_pts .ge. 20) then
                    num20pt(cn)=num20pt(cn)+1
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                else if (num_pts .ge. 12) then
                    num12pt(cn)=num12pt(cn)+1
                endif

c-------------------------------------
c Extract chip name if it is available
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c format: 1 IC3WRIEST;
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
            else if (line(1:1).eq.'1') then
                cell_name(0)=line(3:index(line,';')-1)
                call print_at(10,47,cell_name(cn))
                call tcom(cteol)

c-----------------------------------------------------
c Parse call statements
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c num = cell number being called
c it,im,ir = position of T,M and R in the input line
c cmaxx...cminy represent current cell extents
c delx,dely = translation coordinates
c
c format: C1 MXR0,-1T1000,12000;
c where 1 = cell number being called
c       MX and R0,-1 are optional mirror and rotate
c       commands
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
            else if (line(1:1).eq.'C') then

                if (.not.in_symbol) then   ! when all cells have been defined
                        call push(-100,&99)! and one returns to the top level,
                        in_symbol=.true.   ! in_symbol=.f. Now push -100 to
                endif                      ! indicate this is the top level

                read(line,'(1x,i<index(line,'' '')-2>)',err=96) num
                if (cell_name(num).eq.' ') then
                        error_msg = cell_name(num)//' called before defined'
                        goto 99
                else
                    num_calls(num)=num_calls(num)+1
                    if (levels_below(num)+1.gt.levels_below(cn)) then
                        levels_below(cn)=levels_below(num)+1
                    endif
                    it=index(line,'T')
                    im=index(line,'M')
                    ir=index(line,'R')
                    minp(1)=maxx(num)
                    minp(2)=maxy(num)
                    minp(3)=minx(num)
                    minp(4)=miny(num)
c-----------------------------------------------------
c take care of mirror command
c if it is in the cif line
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c format: C1 MXI1000,5000;
c         C1 MYI4000,50000;
c where MX means mirror x coordinates about the origin
c       MY, mirror y coordinates about the origin
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                    if(im.ne.0) then
                        if (line(im+1:im+1).eq.'X') then
                            call matrix_mult(mtemp,mirrorx,minp,4)
                        else if (line(im+1:im+1).eq.'Y') then
                            call matrix_mult(mtemp,mirrory,minp,4)
                        else
                            error_msg='invalid mirror command'
                            goto 99
                        endif
                    else
                        do i=1,4
                            mtemp(i)=minp(i)
                        enddo
                    endif
c-------------------------------------------------------
c take care of the rotate command
c if it is in the cif line
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c format: C1 MXR0,-1I1000,5000;
c         C1 R-1,0I3000,4000;
c         C1 R0,1I4000,50000;
c where Ri,j takes the coordinate 1,0 to i,j
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                    if (ir.ne.0) then
                        tline=line(ir+1:)
                        if (tline(1:3).eq.'0,1') then
                            call matrix_mult(mout,rot__01,mtemp,4)
                        else if (tline(1:4).eq.'-1,0') then
                            call matrix_mult(mout,rot_m10,mtemp,4)
                        else if (tline(1:4).eq.'0,-1') then
                            call matrix_mult(mout,rot_0m1,mtemp,4)
                        else
                            error_msg='invalid rotate command'
                            goto 99
                        endif
                    else
                        do i=1,4
                            mout(i)=mtemp(i)
                        enddo
                    endif
c-------------------------------------------------------
c take care of the translate command
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
c format: C1 MXR0,-1I1000,5000;
c where Ti,j takes the cell origin and places it at
c       coordinates i,j
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                    if (it.eq.0) then
                        error_msg = 'call without translate not allowed'
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                    goto 99
                endif
                tline=line(it+1:)
                call parse(tline,length-it,points,np,&99)
                call check(tline,length-it,points,2)
                if (np.lt.2) then
                    error_msg = 'invalid translate command'
                    goto 99
                endif
                delx=points(1)
                dely=points(2)
                cmaxx=mout(1)+delx
                cminx=mout(3)+delx
                cmaxy=mout(2)+dely
                cminy=mout(4)+dely
                call update(cn,cmaxx,cmaxy,cminx,cminy)
            endif
        else
            ch=line(1:1)
            if (ch.ne.'(' .and. ch.ne.'L' .and. ch.ne.'E') then
                error_msg = 'syntax error in CIF line'
                goto 99
            endif
        endif
    enddo
10  close(1)

C-----------------------------------------------------------------
C num_calls now contains all the calls made in the top level
C Now push these onto the stack and then push a -100.  This sets
C up the stack to handle the top level and all cells in the same
C manner
C-----------------------------------------------------------------
    do i=1,99
        if (num_calls(i).gt.0) then
            call push(i,&99)
            call push(num_calls(i),&99)
            num_calls(i)=0
        endif
    enddo
    num_calls(0)=1
    call push(-100,&99)

C----------------------------------
C Generate centering report on cells
C----------------------------------
    write(2,*)
    write(2,*) 'Cell Centering Report'
    write(2,*) 'The following cells are not centered'
    write(2,*) 'Cell Name          x error   y error'
    write(2,*) '-----------------------------------'
    do i=0,99
        movex(i) = ((maxx(i)-minx(i))/2 - maxx(i))/100*100
        movey(i) = ((maxy(i)-miny(i))/2 - maxy(i))/100*100
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            ix = movex(i)
            iy = movey(i)
            if (abs(ix).ge.100 .or. abs(iy).ge.100 ) then
                centered=.false.
                write(2,'(a,T18,2i10)') ' '//cell_name(i),ix/100,iy/100
            endif
        enddo
        write(2,*) '-----------------------------------'
        write(2,*) 'End of Cell Centering Report'
        write(2,*)


C-------------------------------
C Generate cell hierarchy report
C-------------------------------
        write(2,*)
        write(2,*) 'Cell Hierarchy Report'
        write(2,*) 'Cell Name    Cells called'
        write(2,*) '-----------------------------------'
        do while(.not.stack_empty)
            call pop(cn)
            cn = -1*(cn+100)
            write(2,*) cell_name(cn)
            fill='----------'
            done=.false.
            do while(.not.done)
                call pop(count)                ! get next #
                if (count.eq.-cn-100) then     ! if it is the same then
                    done=.true.                ! you have finshed this level
                else                           ! otherwise you have count
                    call pop(num)              ! now get cell number
                    write(2,'(1x,a)') fill//'|_ '//cell_name(num)
                    num_calls(num)=num_calls(num)+count*num_calls(cn)
                    fill='          '
                endif
            enddo
        enddo
        write(2,*) '-----------------------------------'
        write(2,*) 'End of Cell Hierarchy Report'
        write(2,*)

C-------------------------------
C Generate cell statistics report
C-------------------------------
19      write(2,*) 'Cell Statistics Report'
        write(2,*) '-----------------------------------'
        do i=0,99
            if (cell_name(i)(1:1).ne.' ') then
                write(2,*) 'Cell Name: '//cell_name(i)
                write(2,*) 'max x dimension: ',maxx(i)/100
                write(2,*) 'min x dimension: ',minx(i)/100
                write(2,*) 'max y dimension: ',maxy(i)/100
                write(2,*) 'min y dimension: ',miny(i)/100
                write(2,*) 'number of times cell called: ',num_calls(i)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            write(2,*) 'number of levels below cell: ',levels_below(i)
            write(2,*) 'number of 12 point polygons: ',num12pt(i)
            write(2,*) 'number of 20 point polygons: ',num20pt(i)
            write(2,*)
        endif
    enddo
    write(2,*) '-----------------------------------'
    write(2,*) 'End of Cell Statistics Report'

C-----------------------------------
C general report on possible errors
C-----------------------------------
    num_12_pt=0
    num_20_pt=0
    do i=1,99
        if (levels_below(i).gt.4) too_many_levels=.true.
        num_12_pt=num_12_pt + num_calls(i)*num12pt(i)
        num_20_pt=num_20_pt + num_calls(i)*num20pt(i)
    enddo

    if (too_many_levels) then
        write(3,*) 'Error:  More than 5 levels in a cell'
    else
        write(3,*) 'Passed: Fewer than 5 levels in design.'
    endif
    write(3,*)
    if (num_12_pt.gt.3500) then
        write(3,*) 'Error:  More than 3500 12 point polygons'
    else
        write(3,*) 'Passed: Fewer than 3501 12 point polygons in design.'
    endif
    write(3,'(9x,a,i<ndig(num_12_pt)>/)')
    1       'Total number of 12 point polygons: ',num_12_pt
    if (num_20_pt.gt.2500) then
        write(3,*) 'Error:  More than 2500 20 point polygons'
    else
        write(3,*) 'Passed: Fewer than 2501 20 point polygons'
    endif
    write(3,'(9x,a,i<ndig(num_20_pt)>/)')
    1       'Total number of 20 point polygons: ',num_20_pt
    if (.not.multiple_of_50) then
        write(3,*) 'Error:  Some vertices not multiples of 50.'
        write(3,*) '        A corrected version can be found in center.cen'
    else
        write(3,*) 'Passed: All vertices multiples of 50'
    endif
    write(3,*)
    if (.not.centered) then
        write(3,*) 'Error:  Some cells not centered.'
        write(3,*) '        A corrected version can be found in center.cen'
    else
        write(3,*) 'Passed: All cells centered'
    endif
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        close(2)
        close(3)
        if (centered .and. multiple_of_50) then
            call tcom(cls)
            stop
        else
C-------------------------------------------------------------
C correct centering errors and errors due to numbers not being
C a multiple of 50 - which corresponds to 0.5 microns
C-------------------------------------------------------------
        open(unit=1,status='UNKNOWN',file=finl,carriagecontrol='list')
        open(unit=4,status='UNKNOWN',file='center.cen',carriagecontrol='list')
        line_num=0
        need_line=.true.
        in_symbol=.false.
        stack_empty=.false.
        call tcom(cls)
        call print_at(10,screen,'Processing Cell:')
        call print_at(12,screen,'Generating Corrected CIF file')

c-----------------------
c start of main read loop
c-----------------------
        do while(.true.)        ! loop terminated by return(1)
                                ! inside getline -> to line 20
            if (need_line) call getline(line,line_num,length,&20)
            need_line=.true.

c-------------------------------------------------
c Define Start - get symbol number and cell name
c-------------------------------------------------
            if (line(1:2).eq.'DS') then
                write(4,'(a)') line(1:length)
                in_symbol = .true.
                read(line,'(3x,i<index(line,'';'')-4>)') cn
                call print_at(10,screen+17,cell_name(cn))
                call tcom(cteol)

c-------------------------------------------------
c Define Finish - restore cell number to top level
c-------------------------------------------------
            else if (line(1:2).eq.'DF') then
                write(4,'(a)') line(1:length)
                in_symbol = .false.
                cn = 0
                call print_at(10,screen+17,cell_name(cn))
                call tcom(cteol)

c---------------------
c Parse box statements
c---------------------
            else if (line(1:1).eq.'B') then
                call parse(line(1:length),length,points,np,&99)
                points(3)=points(3)+movex(cn)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                points(4)=points(4)+movey(cn)

                write(4,'(''B'',4('' '',i<ndig(points(i))>),'';'')')
        1                ,(m50(points(i)),i=1,4)


c-------------------------
c Parse polygon statements
c-------------------------
            else if (line(1:1).eq.'P') then
                done=.false.
                num_pts=0
                do while(.not.done)
                    call parse(line(1:length),length,points,np,&99)
                    do i=1,np,2
                        points(i)=points(i)+movex(cn)
                        points(i+1)=points(i+1)+movey(cn)
                    enddo
                    if (line(1:1).eq.'P' .and. index(line,';').gt.0.and.
        1                       index(line,';').le.length) then
                        write(4,'(''P'',<np/2>('' '',i<ndig(points(i))>,'',''
        1                                     ,i<ndig(points(i+1))>),'';'')')
        1                ,(m50(points(i)),m50(points(i+1)),i=1,np,2)
                    else if (line(1:1).eq.'P') then
                        write(4,'(''P'',<np/2>('' '',i<ndig(points(i))>,'',''
        1                                      ,i<ndig(points(i+1))>))')
        1                ,(m50(points(i)),m50(points(i+1)),i=1,np,2)
                    else if (index(line,';').gt.0.and.
        1                       index(line,';').le.length) then
                        write(4,'(<np/2>('' '',i<ndig(points(i))>,'',''
        1                                     ,i<ndig(points(i+1))>),'';'')')
        1                ,(m50(points(i)),m50(points(i+1)),i=1,np,2)
                    else
                        write(4,'(<np/2>('' '',i<ndig(points(i))>,'',''
        1                                     ,i<ndig(points(i+1))>))')
        1                ,(m50(points(i)),m50(points(i+1)),i=1,np,2)
                    endif
                    i=index(line,';')
                    if (i.ge.1 .and. i.le.length) then
                        done=.true.
                    else
                        call getline(line,line_num,length,&97)
                    endif
                enddo

c----------------------
c Parse call statements
c----------------------
            else if (line(1:1).eq.'C') then

                read(line,'(1x,i<index(line,'' '')-2>)') num
                it=index(line,'T')
                im=index(line,'M')
                ir=index(line,'R')
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                tline=' '
                len=1

                minp(1)=movex(num)
                minp(2)=movey(num)
                minp(3)=0
                minp(4)=0

c----------------------------
c take care of mirror command
c----------------------------
                    if(im.ne.0) then
                        tline=' '//line(im:im+1)
                        len=3
                        if (line(im+1:im+1).eq.'X') then
                            call matrix_mult(mtemp,del_mirx,minp,4)
                        else if (line(im+1:im+1).eq.'Y') then
                            call matrix_mult(mtemp,del_miry,minp,4)
                        endif
                    else
                        do i=1,4
                            mtemp(i)=minp(i)
                        enddo
                    endif

c--------------------------------
c take care of the rotate command
c--------------------------------
                    if (ir.ne.0) then
                        if (line(ir+1:ir+3).eq.'0,1') then
                            tline=tline(1:len)//line(ir:ir+3)
                            len=len+4
                            call matrix_mult(mout,del__r01,mtemp,4)
                        else if (line(ir+1:ir+4).eq.'-1,0') then
                            tline=tline(1:len)//line(ir:ir+4)
                            len=len+5
                            call matrix_mult(mout,del_rm10,mtemp,4)
                        else if (line(ir+1:ir+4).eq.'0,-1') then
                            tline=tline(1:len)//line(ir:ir+4)
                            len=len+5
                            call matrix_mult(mout,del_r0m1,mtemp,4)
                        endif
                    else
                        do i=1,4
                            mout(i)=mtemp(i)
                        enddo
                    endif

c-----------------------------------
c take care of the translate command
c-----------------------------------
                tline=tline(1:len)//'T'
                call parse(line(it+1:),length-it,points,np,&99)
                points(1)=points(1)+movex(cn)-mout(1)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            points(2)=points(2)+movey(cn)-mout(2)

            write(4,'(''C'',i<ndig(num)>,a,i<ndig(points(1))>,'','',''
     1                ,i<ndig(points(2))>,'';'')')
     1                ,num,tline(1:len+1),(m50(points(i)),i=1,2)

        else
            write(4,'(a)') line(1:length)
        endif
    enddo
20  close(1)
    close(4)

        call tcom(cls)
        stop
    endif

C-----------------------
C Error report generation
C-----------------------
96  error_msg='invalid format in call command'
    goto 99
97  error_msg='unexpected end of input'
99  write(3,*)
    write(3,*) 'Center has terminated due to errors'
    write(3,*)
    write(3,*) error_msg
    write(3,*)
    write(3,*) line(1:length)
    write(3,*)
    write(3,*) 'error occured in line: ',line_num
    write(3,*) 'current cell: ',cell_name(cn)
    write(3,*)
    write(3,*) 'See the message file for statistics obtained prior to'
    write(3,*) 'the occurence of the error.'
    write(3,*)
    goto 19
    end

c-----------------------------------------------
c get a line of input from the cif data file
c if end of file is reached, take second return
c if no semicolon found take first return
c otherwise return as normal
c-----------------------------------------------
    subroutine getline(line,line_num,1,*)
    character*80 line,error_msg
    logical no_semic
    common /error/error_msg,no_semic,multiple_of_50
    write(line,'(80('' ''))')
    read(1,'(q,a)',end=2) 1,line(1:1)
    line_num=line_num+1
    i=index(line,';')
    if (i.lt.1 .or. i.gt.1) then
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        common /stack/stack,sp,stack_empty
        common /error/error_msg,no_semic,multiple_of_50
        if (sp.lt.0) then
                stack_empty=.true.
                error_msg = 'stack underflow'
                return
        endif
        i = stack(sp)
        sp=sp-1
        if (sp.lt.0) then
                stack_empty=.true.
        else
                stack_empty=.false.
        endif
        return
        end

C-----------------------------------------
C find first numeric character in input line
C-----------------------------------------
        integer function ifn(line,l,i)
        character*(*) line
        logical done
        done=.false.
        ic = ichar(line(i:i))
        if(ic.ge.48 .and. ic.le.57) done=.true.
        do while (.not.done)
            i=i+1
            ic = ichar(line(i:i))
            if(i.gt.1 .or. (ic.ge.48 .and. ic.le.57)) done=.true.
        enddo
        if (i.gt.1 .and. line(i-1:i-1).eq.'-') i=i-1
        ifn = i
        return
        end

C-----------------------------------------
C find first non numeric character in input line
C-----------------------------------------
        integer function ifnn(line,l,i)
        character*(*) line
        logical done
        done=.false.
        if (line(i:i) .eq. '-') i=i+1
        ic = ichar(line(i:i))
        if (ic.lt.48 .or. ic.gt.57) done=.true.
        do while (.not.done)
            i=i+1
            ic = ichar(line(i:i))
            if (i.gt.1 .or. ic.lt.48 .or. ic.gt.57) done=.true.
        enddo
        ifnn = i
        return
        end
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
C-----------------------------------------------
C This routine takes all numeric values in the
C input line and places them in the array points
C-----------------------------------------------
        subroutine parse(line,l,points,numpts,*)
        integer points(*),fn,nn,numpts,pos
        character*80 error_msg
        logical done,multiple_of_50,no_semic
        character*(*) line
        common /error/error_msg,no_semic,multiple_of_50
        numpts=0
        done=.false.
        pos=1
        do while(.not.done)
            fn=ifn(line,l,l*pos)
            nn=ifnn(line,l,l*fn)
            if (fn.le.l) then
                numpts=numpts+1
                read(line(fn:),'(i<nn-fn>)',err=1) points(numpts)
                pos=nn
            else
                done=.true.
            endif
            if (nn.gt.l) done=.true.
        enddo
        return
1       error_msg = 'syntax error in CIF line, unable to read coordinates'
        return(1)
        end

        subroutine check(line,length,points,np)
        character*80 error_msg,line,cell_name(0:99)*10
        integer points(*)
        integer maxx(0:99),maxy(0:99),minx(0:99),miny(0:99)
        integer num_calls(0:99),cn
        logical multiple_of_50,no_semic
        common /info/cell_name,maxx,maxy,minx,miny,num_calls,cn,line_num
        common /error/error_msg,no_semic,multiple_of_50

        do i=l,np
            if (points(i)/50*50 .ne. points(i)) then
                multiple_of_50=.false.
                write(2,*) 'not multiples of 50, cell: '//cell_name(cn)//
       1                        ' line number:',line_num
                write(2,*) line(l:length)
                write(2,*)
            endif
        enddo
        return
        end

C---------------------------------------
C this routine calculates how many digits
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
C are needed when writing i
C--------------------------------------
        integer function ndig(i)
        if (i.eq.0) then
            ndig=1
            return
        else if (i.lt.0) then
            len=2
        else
            len=1
        endif
        ndig=len+alog10(float(abs(i)))
        return
        end

C-------------------------------------------------------------------------
C the input matrix, minp is multiplied by the transform matrix, mtrans
C to generate the output matrix, mout.  This routine is used by the call
C sections to handle the mirror and rotate commands
C-------------------------------------------------------------------------
        subroutine matrix_mult(mout,mtrans,minp,size)
        integer mout(4),mtrans(4,4),minp(4),size,row,col
        do row=1,size
          mout(row)=0
          do col=1,size
            mout(row)=mout(row)+mtrans(row,col)*minp(col)
          enddo
        enddo
        return
        end
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Vita Auctoris

Peter Bird was born on October 29, 1963 in Hamilton, Ontario. He completed his high school education at Sandwich Secondary School in Windsor in 1981. He graduated from the University of Windsor in 1985 with a Bachelor of Applied Science in Electrical Engineering and was a recipient of the 1984 Digital Equipment of Canada Award of Merit. In May 1987 he finished his Masters of Applied Science in Electrical Engineering, also at the University of Windsor. He is currently with Northern Telecom in Ottawa working in the New Product Introduction Department of the Silicon Compontents Group. Peter Bird is a member of the IEEE.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.